COBOL 言語の評価と展望

Evaluation and View of COBOL Language

現在，企業では市場変化への迅速な対応を行うための次世代 IT インフラへの移行が進んでいる。そのなかで，現在の情報システムの主要部分を担う，COBOL 言語で記述された膨大な資産に関する移行の問題が注目されている。ここでは，生産性，保守性，機能性の観点での COBOL 言語の分析を通して，ビジネスロジックの記述言語として COBOL 言語に優位性があると評価する。また，今後も企業のビジネス基盤を支え続けるために COBOL 言語に求められるソリューションを検討し，「標準化」「可視化」「モデリング」を軸にしたソリューションを提案する。

佐々木　仁　Sasaki Jin
前田　光司　Maeta Kouji
庄司　秀明　Shouji Hideaki
及川　清太郎　Oikawa Seitaro
加賀　雅弘　Kaga Masahiro

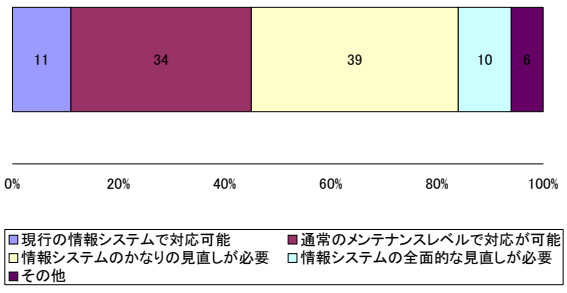
１．はじめに

近年，社会・産業構造の急激な変化にともない，市場変化への迅速な対応が，企業のビジネス戦略上の重要な課題となっている。その一環として，次世代 IT インフラへの移行を目的とした投資が本格化している。次世代 IT インフラへの移行は，「レガシーマイグレーション」という名称で重要性が広く認知されている。

現在，日本の企業内で稼動している多くの情報システムは，メインフレームシステムを中心に構築されている。社団法人日本情報システム・ユーザー協会が実施した企業 IT 動向調査 2007（以下，企業 IT 動向調査 2007 と記す）によると，日本企業の情報システムのメインフレーム比率は 50%前後の高い割合を占めている [1]。しかし，オープン系ハードウェアの価格性能比の向上により，これまでメインフレーム以外では実現が困難であった高信頼・高性能のシステムが，オープンシステムで実現可能となっていることから，メインフレームシステムのオープン化が活発化している。

一方，メインフレーム上で稼動している情報システムのソフトウェア的な側面では，企業 IT 動向調査 2007（図 1 参照）によると「通常メンテナンスレベルで対応可能」が 34%，「かなりの見直しが必要」が 39%と，現在の情報システムに対するユーザーの認識は完全に二極化している。また，この比率に応じた IT 投資予算を確保している企業は少なく，「霧の中での必要予算の策定」（社団法人日本情報システム・ユーザー協会）と分析されるなど，現在の情報システムに対するユーザーの認識が揺らいでいる。

この背景には，メインフレーム上の情報システムの多くが COBOL 言語で記述されており，10 年以上にわたる保守・運用を経ていまなお安定稼動している事実がある。いま，COBOL 言語で記述された情報システム（以下，COBOL 資産と記す）が，企業のビジネス戦略上の重要な課題として，再び注目されている。



「企業 IT 動向調査 2007」のデータをもとに作成

図 1　必要と考えられる情報システムの対応

２．COBOL 資産を取り巻く環境

2.1 COBOL 資産の現状

日本の主要ベンダーで構成されている非営利団体 COBOL コンソーシアムによると，現在，日本の大手企業が保有する COBOL 資産は 1000 万行から 1 億行，COBOL 資産を支える国内の COBOL 技術者人口は 50 万人ともいわれている [2]。しかし，2007 年問題に付随し

て，メインフレーム上の情報システムを長年にわたり支えてきた COBOL 技術者が 2007 年を境に減少に転じること，また，若手 IT 技術者は C++／Java 技術者がその大半を占めていることから，技術の二極化が進んでいる。2007 年問題や技術の二極化が進むことで，COBOL 技術者人口は 2007 年を境に大幅に減少すると予想されている。

COBOL 技術者の大幅な減少により，これまで長年にわたり安定稼動してきたメインフレーム上の COBOL 資産の継承が困難となってきており，レガシーマイグレーションとして COBOL 資産をそのまま継承するか，または Java などの他の言語へ移行するなどの見直しを行うかの選択に迫られている。

2.2 最新技術動向

メインフレーム上の情報システムの移行について，最近の技術動向としては，COBOL 資産の一部を見直し，Java や XML など最新技術と連携する Web アプリケーションとして再構築するニーズが多い。Web アプリケーションへの再構築に対応するためには，既存の COBOL 資産と Java／XML などの最新技術と連携するインタフェースが必要となる。

現在のオープン系 COBOL コンパイラでは Java／XML などの最新技術との連携機能をサポートしており，COBOL 資産との連携を容易に行うことができる。例えば，日立のオープン系 COBOL コンパイラ「COBOL2002」では，表 1 の連携機能を標準でサポートしている [3)]。

表 1　COBOL2002 の Web システム連携機能

Java 連携機能	Java プログラムと COBOL プログラムとの連携
XML 連携機能	e ビジネス向けのデータ交換用 XML データを扱うアプリケーションを作成
SOAP 連携機能	Microsoft 社の.NET Framework を基盤とした Web サービスプログラムの作成を支援

2.3 COBOL 資産の評価に関する問題点

COBOL 資産の継承に際して，COBOL 技術者の減少という課題はあるものの，COBOL 資産の継承を支えるオープン系 COBOL コンパイラは市場ニーズを満たす機能を十分にサポートしている。

しかし，1 章で述べたとおり，企業による COBOL 資産の評価は二極化しており，COBOL 資産の移行に関しては否定的な見解も多い。要因としては，情報システムの長年の安定稼動にともない，社会インフラの背骨としてあまりに基本的すぎて，COBOL 資産の特性の十分な分析と評価がなされておらず，ユーザーの判断材料に乏しいことが挙げられる。市場変化へ迅速に対応できる次世代インフラを構築するためには，現在の情報システムの主要部分を担う COBOL 資産の分析・評価を十分に行う必要がある。

3．COBOL の言語特性の分析

3.1 言語特性

市場変化への迅速な対応を可能とする次世代 IT インフラへの移行の中で，現在の情報システムの分析・評価を行う重要性が増してきている。なかでも現在の情報システムの主要部分を担う COBOL 資産の分析・評価が重要である。

（株）日立東日本ソリューションズ（日立ＴＯ）は，2002 年より，（株）日立製作所ソフトウェア事業部で開発している，第 4 次 COBOL 言語規格に準拠したオープン系 COBOL コンパイラ「COBOL2002」の開発に参画している。本章では，開発を通じて蓄積した COBOL 言語に関する技術をもとに，「生産性」「保守性」および「機能性」の観点で，C 言語／C++, Java との比較を通じて，COBOL 資産の分析・評価を行う。

3.2 分析

3.2.1 生産性

図 2 に生産効率（ks／人月），および不良密度（件／ks）の，C 言語／C++を基準（100%）とした相対グラフを示す。C 言語／C++はシステム記述言語としての側面が強く，あくまで参考値であるが，COBOL 言語は C 言語／C++や Java に比べ，生産効率が２～３割程度高いのに加え，不良密度も C 言語／C++に比べ３～４割程度小さい。

COBOL 言語では，統合開発マネージャ，テストデバッグ環境およびテスト支援ツール，ならびに可読性の高い言語構文を生かしたプログラム自動生成ツールなどの開発環境の充実度が高いことが生産性の高さの要因として挙げられる。オフショア開発などでも技術者のスキルによらず安定した品質を確保しやすい言語といえる。

3.2.2 保守性

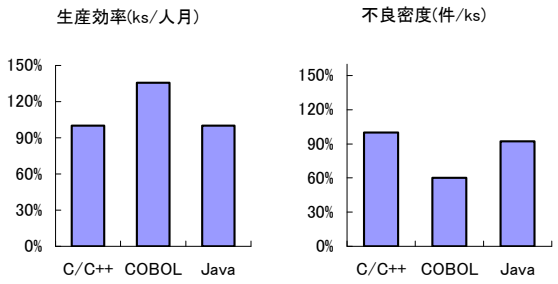


図 2　生産効率と不良密度

(1) 安定稼動実績

COBOL 言語は，信頼性が高度に要求される基幹業務での 30 年以上にわたる稼動実績により，安定性は十分に実証されている。安定稼動実績の点では，COBOL 言語は C 言語／C++や Java に対して，大きな優位性がある。しかし，システムの安定性は適用される分野や業務に依存すること，また，C 言語/C++や Java の適用事例も今後増加していくと考えられることから，将来的な視点での安定性は同等といえる。

(2) プラットフォーム依存性

C 言語／C++はシステム記述言語としての側面が強くプラットフォーム依存性が高い。一方，Java と COBOL 言語はプラットフォームに依存しないよう設計された言語である。このため，メインフレーム上の COBOL 資産のオープン系システムへの移行も容易である。表 2 に，日立のオープン系 COBOL コンパイラの対応プラットフォームを示す。すでにオープン系の主要なプラットフォームへ対応しており，現在も新規プラットフォームに対応した製品を開発中である。

表 2　対応プラットフォーム

OS	プラットフォーム
Windows	Windows(x86)
	Windows(x64)
Linux	Linux(x86)
	Linux(IPF※1)
AIX	AIX 5L(Power)
HP-UX	HP-UX(PA-RISC)
	HP-UX(IPF※1)
Solaris	Solaris(SPARC)

※1 Itanium Processor Family

(3) 開発方法論

表 3 にそれぞれの言語を採用した場合の開発方法論を示す。これまで蓄積した資産の継承という点で，旧来の開発方法論である構造化設計手法を採用できる C 言語／C++と COBOL 言語に優位性がある。しかし，今後ユーザーサイトでは，オブジェクト指向設計が主流となる可能性も高い。COBOL 言語では第 4 次 COBOL 言語規格でオブジェクト指向機能が追加され，日立オープン系 COBOL コンパイラ「COBOL2002」でオブジェクト指向機能をサポートしている。したがって，将来的な視点では，それぞれの言語の開発方法論という観点での評価は同等である。しかし，Java では多機能なクラスライブラリの拡張にともない，JSP，EJB など開発方法論自体も変化しており，拡張性が非常に高い反面，常に最新のクラスライブラリの技術習得が技術者に要求される。

以上，保守性の 3 つの観点を総合的に見た場合，ビジネスロジック記述言語としては COBOL 言語と Java は同等であり，それぞれの得意とする分野と業務ロジックの親和性によって優劣が決まるといえる。

表 3　各言語の開発方法論

	C／C++	COBOL	Java
構造化設計	可能	可能	不可能
オブジェクト指向設計	可能	可能	可能

3.2.3 機能性

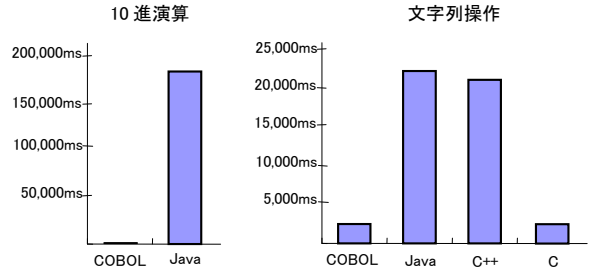
C 言語／C++と Java は，言語構文においてほぼ同等の機能をもつ。ビジネスロジックでの Java の優位性は，Web アプリケーションなどとの親和性が高い多機能なクラスライブラリが提供されている点である。それに対して，COBOL 言語は，言語構文そのものからビジネスロジックに特化された体系をもつ。このような特性をもつ COBOL 言語と同等な機能を，Java などで記述する場合，ロジックが複雑となり保守性が低下する。

ビジネスロジックに多用される文字列操作と 10 進演算 [2]の性能比較結果を図 3 にまとめる。10 進演算は四則演算およびその複合演算を 10 万回繰り返した結果であり，文字列操作は文字列の計数，置換，連結，分割を 10 万回繰り返した結果である。

Java と C++では，10 進演算および文字列操作に標準的なクラスライブラリを使用している。C 言語との比較でもわかるとおり，クラスライブラリを使用した場合，性能上の負荷が大きい。10 進演算および文字列操作を標

準構文で実現できる COBOL 言語は，高速処理が可能である。なお，C 言語／C++には 10 進演算を行う機能はなく，整数型および浮動小数点型を使用した演算では，COBOL 言語と同等の演算結果の精度を保証できないため比較の対象外としている。

図 3 から，ビジネスロジックに限定した場合，COBOL 言語に性能的優位性があり，現在の COBOL 資産を単純に C 言語／C++や Java などに単純に移行できないことがわかる。COBOL 言語の性能上の優位性によって，COBOL 資産を他の開発言語へ移行した場合，バッチ処理などで，性能要件を満たせないことがありえる。



動作環境
Windows 2000 Professional(SP4),CPU:2.4GHz,メモリ 1GB
COBOL 動作環境:日立 COBOL2002 01-03
C/C++動作環境:Visual Studio 2005
Java 動作環境:JDK 5.0

図 3　10 進演算と文字列操作の性能比較

3.3 評価

3.2 節での分析結果を表 4 にまとめる。市場変化へ柔軟に対応できるシステムを構築するうえで，拡張性は重要な要素である。Java は，プラットフォーム依存性が少なく，拡張性に優れたクラスライブラリをもつ有望な開発言語である。しかし，Java の拡張性はクラスライブラリの拡張に依存し，拡張されたクラスライブラリを使用するためには，Java 技術者の継続的な育成をともなう点に留意しなければならない。Java は，COBOL 言語と同様に，技術者不足の問題を潜在的に抱えているのである。

本章で分析したように，COBOL の言語特性は，ビジネスロジックを記述するのに適している。また，COBOL の言語規格は，現在の COBOL 資産を継承しつつ，次世代の技術および開発方法論に適合するよう設計された規格である。2002 年に制定された第 4 次 COBOL 言語規格のオブジェクト指向機能では，現在の COBOL 資産のビジネスロジックを継承したまま，部分的な修正のみで，オブジェクト指向の開発手法に切り替えることが可能である。さらに，最新のオープン系 COBOL コンパイラは，Java や XML など最新技術連携機能を備えており，拡張性も十分である。

以上の分析により，COBOL 言語は今後もビジネスロジック記述言語として採用していくのに十分な機能をもち，市場変化への迅速な対応を可能とするインタフェースをもっていると評価できる。

表 4　ビジネスロジックへの適合性の評価

	C／C++	COBOL	Java
生産性	普通	高い	普通
保守性	低い	普通	普通
機能性	低い	高い	普通

４．今後の展望

4.1 ソリューションの検討

レガシーマイグレーションの流れの中では，2007 年問題により，COBOL 資産の継承はネガティブな選択肢として捉えられることが多い。しかし，現実には膨大な COBOL 資産が現在のオープン環境へ移行されており，企業のビジネス基盤を支えていくためには COBOL 資産の継承を積極的に捉えていく必要がある。また，現在稼動している情報システムの主要部分を担う COBOL 資産の分析・評価を通じ，COBOL 言語はビジネスロジックの記述言語としては他の言語に対して優位性があることを改めて評価した。本章では，COBOL 資産の継承を積極的に捉えなおし，市場変化へ迅速に対応できる次世代 IT インフラを支え続けるために，必要となるソリューションを検討する。

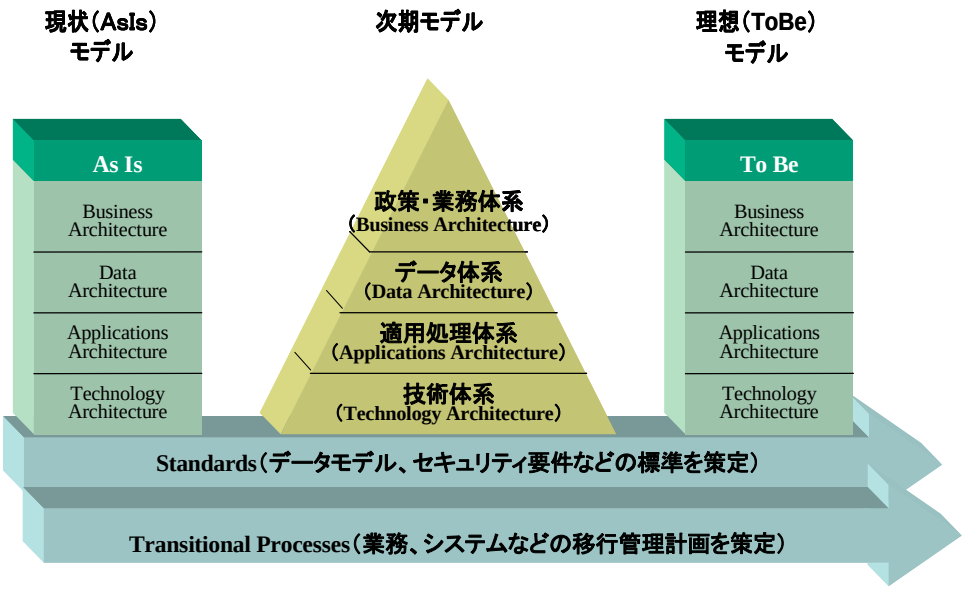


図 4　EA のフレームワーク

検討のフレームワークとして，エンタープライズ・アーキテクチャー（以下，EA と記す）の概念を使用する。EA は，特定の技術や特定のビジネスモデルによらず，「全体最適」の観点から市場変化に迅速に対応できるビジネスプロセスを体系化するフレームワークとして，近年注目を集めている。EA のフレームワークは，現在，企業が抱える課題を体系的に整理し，今後の改善プロセスを体系的にまとめたものである [4]。したがって，EA の目的はレガシーマイグレーションの目的と同じであり，より体系的に次世代 IT インフラを支えるソリューションを検討するうえで，最適なフレームワークである。

4.2 エンタープライズ・アーキテクチャー（EA）

図 4 に EA の概念図を示す。本節では EA の概要を説明する。EA では，常に時代の変化に迅速に対応できるビジネスプロセスを，循環的な改善プロセスの体系として定義する。改善プロセスは，現行（AsIs）モデルの分析にもとづき，理想（ToBe）モデルの策定，および次期ビジネスモデルへの移行計画の策定を循環的に実施するスパイラルな構造をもつ。各モデルは以下の 4 つの層に分割される。

(1)「政策・業務体系」
(2)「データ体系」
(3)「適用処理体系」
(4)「技術体系」

最上位層である「政策・業務体系」から最下位層である「技術体系」までの各層を，トップダウンで段階的に詳細化・モデリングすることで，ビジネスモデルと IT 技術の透過的な連動を可能にし，市場変化への対応力を向上させることを目的としている。

4.3 潜在的なユーザーニーズ

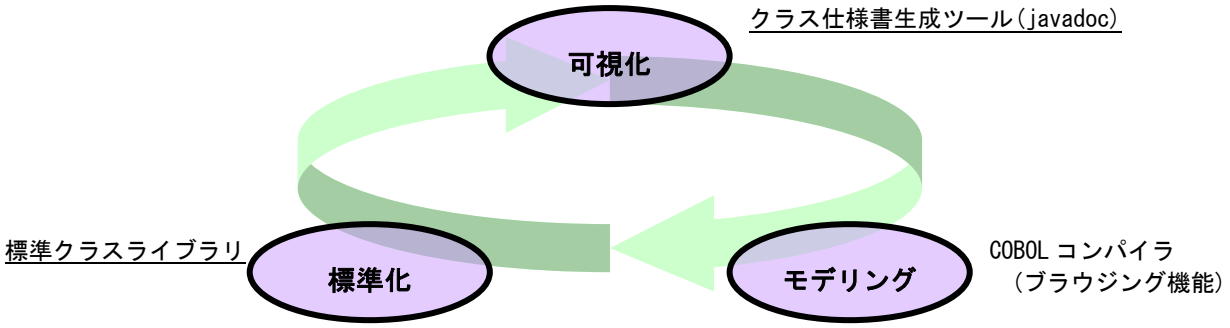
EA の概念で重要なのは「循環的な改善プロセス」の確立である（いまの To Be は未来の As Is）。循環的な改善プロセスの確立のためには，「可視化」「モデリング」「標準化」を循環的に行うことが重要となる。

3 章では，これまでの開発方法論にもとづく観点にしたがって分析を行い，ビジネスロジック記述言語としての COBOL 言語の優位性が非常に高いことを改めて評価した。しかし，今後は EA の重要概念である「可視化」「モデリング」「標準化」の観点でのニーズが高まると予想され，これらの観点での潜在的なニーズを明確化することが COBOL 資産の展望を見るうえで重要となる。

EA の重要概念「可視化」「モデリング」「標準化」を軸として COBOL 資産に対して提供されている現在のソリューション（サービスまたは製品（機能））の一例を図 5 にまとめる。図 5 において下線で示した C／C++, Java でのみ提供されているソリューションがあることからわかるように，今後重要度が増すと考えられる「可視化」「モデリング」「標準化」のニーズに対して，COBOL 言語はまだ発展の余地がある。

4.4 COBOL 言語の展望

本章では，COBOL 資産の継承を積極的な選択肢として捉えなおし，今後も企業のビジネス基盤を支えつづけるために必要なソリューションの検討を行った。EA の



下線は，C 言語/C++, Java でのみ提供されているサービス・製品(機能)を示す。

図 5　「可視化」「モデリング」「標準化」を軸にしたソリューション

フレームワークをもとに「可視化」「モデリング」「標準化」を軸に展開される COBOL 関連のソリューションにはまだ発展の余地があり，今後のユーザーニーズに答えるためにはソリューションの拡充が必要である。

日立ＴＯでは，COBOL，Fortran，C 言語などのコンパイラ開発技術，C++クラスライブラリの開発 [5]，C++開発支援ツール [6]などの開発を通じて蓄積された技術をもとに以下のようなソリューションを検討中である。今後，検討内容に応じて，日立ＴＯとしてのソリューション開発，ソフトウェア事業部との共同でのソリューションや製品機能の開発など，ユーザーニーズに即した最適なソリューションの提案を行っていく所存である。

(1) COBOL 設計仕様書作成支援ツールの整備

構築されたシステムは，迅速に「可視化」する必要がある。COBOL 言語は，一般文書の構造に近い言語構造を採用することで，高い可読性を実現するように設計された言語である。すなわち，COBOL 言語の構造(プログラム，節，段落)は，そのまま設計仕様書の論理構造(章，節，段落)に一致させることが可能である。しかし，COBOL 言語の構文体系が英語文化圏を想定しているため，日本においては COBOL 言語の可読性は十分に活用されていない。COBOL 言語の可読性を有効に生かすため，設計仕様書と COBOL プログラムを相互に変換するツール群を整備することで，ユーザーサイトでの可視化を支援する。

(2) COBOL ブラウジング機能の拡充

現行システムの見直し，または次期システムを構築するうえで，迅速に「モデリング」を行うことが重要となる。COBOL 言語のプログラム・メソッド呼び出し（CALL 文，INVOKE 文）や節（SECTION）のモジュール階層および関連図だけでなく，ファイル入出力や SQL アクセス，XML 連携機能など，個々のプログラム特性をモデリングするツール群を整備することで，ユーザーサイトでの現行システムの分析を支援する。

(3) 標準クラスライブラリの整備

変化に迅速に対応するようモデリングされたシステムを構築するうえで，EA 各層は「標準化」されていることが重要となる。標準化を支援するソリューションとしては，例えば，EA の成果物であるサービスコンポーネント参照モデル（適用処理体系を実現するための雛形となるモデル）に準拠した，COBOL 言語のクラスライブラリの整備がある。

Java のクラスライブラリの拡張にともなう継続的な技術者育成の問題を回避するために，COBOL 言語が整備するべきクラスライブラリとしては，業務コンポーネントごとに，使用頻度の高い標準的な業務処理をメソッドとしてまとめたクラス体系が最適といえる。このクラスライブラリを使うことで，ユーザーはこれまでの COBOL 資産のビジネスロジックを継承しつつ，メソッドとして提供される標準的な業務処理へと，段階的に移行を行うことが可能となり，COBOL 資産の標準化を支援することができる。

５．おわりに

レガシーマイグレーションでは，これまでの日本の情報システムを安定的に支えてきた COBOL 資産の移行は重要な鍵である。今後も企業の安定したビジネス基盤を確保するために COBOL 言語へのユーザーからの期待は大きい。日立ＴＯでは，COBOL 言語に対するユーザー

ニーズに対して COBOL 言語の周辺機能の拡充を続けるとともに，最適なレガシーマイグレーションの解としてのソリューションの検討および提供を行っていきたいと考える。

参考文献

１）社団法人日本情報システム・ユーザー協会
http://www.juas.or.jp/index.html
２）COBOL コンソーシアム
http://www.cobol.gr.jp/
３）（株）日立製作所，COBOL 開発環境（COBOL2002 ファミリー）
http://www.hitachi.co.jp/Prod/comp/soft1/cobol/
４）経済産業省，EA ポータル
http://www.meti.go.jp/policy/it_policy/ea/index.html
５）新藤南平 他：生産性向上のための C++実装設計，日立ＴＯ技報 第 5 号，1999 年
６）山内健一 他：プラットフォーム移行診断サービスの実績と今後の展開，日立ＴＯ技報 第 10 号，2004 年

佐々木 仁 2001 年入社
UB ビジネスソフト開発グループ
COBOL2002 コンパイラ開発
ji_sasaki@hitachi-to.co.jp

前田 光司 1988 年入社
UB ビジネスソフト開発グループ
COBOL 連携製品の開発
maeta@hitachi-to.co.jp

庄司 秀明 1990 年入社
UB ミドルウェア開発グループ
COBOL2002 コンパイラ開発
hideaki@hitachi-to.co.jp

及川 清太郎 2004 年入社
UB ビジネスソフト開発グループ
COBOL2002 コンパイラ開発
s_oikawa@hitachi-to.co.jp

加賀 雅弘 1992 年入社
UB ビジネスソフト開発グループ
COBOL2002 コンパイラ開発
kaga@hitachi-to.co.jp