Web対応事務処理スクリプト言語「COBOLスクリプト」

三宅立記[†] 今城哲二[††,†††] 佐藤忍[†]
井藤敏之[†] 横塚大典[††]
辻畑好秀[††] 植村俊亮[†††]

既存の企業情報システムを安価に再構築するインフラとしてWebを利用するイントラネットの導入が急速に進行している．しかし，企業情報システムのイントラネット化には，システム開発の面で次のような問題がある．(1) システム開発を行うための技術としてHTML，Java等をシステム開発者が新たに習得する必要がある．(2) COBOL，PL/I等の正確な精度を保証する10進演算機構が利用できないため，金額処理を伴ったシステム開発が困難．(3) 従来のクライアントサーバシステムや端末に比べてレスポンスが悪い．

COBOLスクリプトは，これらの問題を解決するために日立製作所と日立ソフトウェアエンジニアリングが共同開発したスクリプト言語であり，次のような特長を有する．(1) 企業内の情報システム部門で最もよく利用されているCOBOL85をベースにWebに必要な機能に絞り込んだ言語仕様．(2) メインフレーム系のCOBOLと同一精度の10進演算機能をサポート．(3) 既存のCOBOL処理系の長所，短所の分析に基づいた効率のよい実装．

これにより次の成果を得た．(1) 金額計算など誤差の許されない分野の業務をWeb上で心配なく稼動可能とした．(2) テストデバッガやカバレージのサポートにより大規模な開発プロジェクトでの利用を可能にした．(3) プログラムの保守性を向上する日本語プログラミング機能をサポートした．(4) 優れた実行性能を実現できた．

COBOL Script: A Business-oriented Scripting Language

Tatsuki Miyake,[†] Tetsuji Imajo,[††,†††] Shinobu Sato,[†]
Toshiyuki Ito,[†] Daisuke Yokotsuka,[††] Yoshihide Tsujihata[††]
and Shunsuke Uemura[†††]

The introduction of intranet which allows to access Web technology is rapidly growing as the infrastructure to restructure existing business information systems within a reasonable cost. However, in terms of system development, use of intranet on the business systems poses the following problems. (1) The system developers must learn new technology such as HTML and Java as the technology for system development. (2) As the decimal arithmetic functions, which guarantee the precision such as in COBOL and PL/I, are not available, it is difficult to develop systems that involves accounting processing. (3) The response time might exceed those of the existing Client and Server systems or terminals.

COBOL Script, which is a script language developed jointly by Hitachi Ltd. and Hitachi Software Engineering Co., Ltd. to solve these problems, has the following features. (1) The language specification, which consists of required functions for Web computing, is a subset of COBOL85, which is the most frequently used programming language in business information systems. (2) The decimal arithmetic functions with the same precision as in the standard COBOL85 on main-frame computers are supported. (3) Efficient implementation is based on analysis of the pros and the cons of the COBOL processing system.

We obtain the following results; (1) Applications requiring high precision such as accounting processing can be operated on Web. (2) The test debugger and the coverage functions make it possible to use in a large development project. (3) The Japanese programming facility provides good maintainability. (4) Good performance is achieved.

† 日立ソフトウェアエンジニアリング（株）開発事業部
Development Division, Hitachi Software Engineering Co., Ltd.
†† （株）日立製作所ソフトウェア事業部
Software Division, Hitachi, Ltd.
††† 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information science, Nara Institute of Science and Technology

1. はじめに

PC及びネットワークの急速な普及に伴い，インターネットの利用者も急激に増加している．特に不特定多数の利用者向けに安価で手軽に情報発信する手段としてWebが最も注目を集めており，その理由は次の通りである．

(1) 基本的な仕掛けがシンプルでシステム構築及び利用共に容易である．
(2) サーバ，ブラウザをはじめとして，いわゆるフリーソフトが充実しており，システム構築及び利用の両面で非常に安価である．
(3) 利用者がコンテンツをダウンロードする形態のため，不特定多数の利用に向いている（配信の問題が少ない）．

利用者の急速な増加と共に技術開発競争も激化し，HTML[1]の表現力や能力を超えるXML，Java[2]，スクリプト言語等が次々と開発され市場に投入されてきている．この中でスクリプト言語は，HTMLの機能を拡張する最も安易な手段であり，その特徴は次の通りである．

(1) HTMLの一部としてプログラムを埋め込むことにより，ブラウザの機能やHTMLで記述した内容をオブジェクトとして取り扱うことができ，これらをプログラム中で操作できる．これにより，ホームページの作成者は通常のHTMLよりも強力なGUIを利用者に提供することができる．
(2) Webサーバが持つCGI[3]等のインタフェースにより，サーバ上で動作するプログラムを記述することができる．

Webで利用可能でブラウザと連動するスクリプト言語としては，ブラウザ市場を2分するNetscape社Navigator及びMicrosoft社Internet Explorerが提供するJavaスクリプトがもっとも一般的である．さらに，Microsoft社は，Basicプログラマ向けにVisual Basicスクリプト[4]も提供している．一方，サーバ上で動作するスクリプト言語としては，前述のJavaスクリプトとVisual Basicスクリプト以外にPerlがあり利用者も多い．

これらを背景として既存の企業情報システムを安価に再構築するインフラとしてWebを利用するイントラネットの導入も急速に進行している．しかし，企業情報システムのイントラネット化には，システム開発の面で次のような問題がある．

(1) システム開発を行うための技術としてHTML，Java等をシステム開発者が新たに習得する必要がある．
(2) COBOL，PL/I等の正確な精度を保証する10進演算機構を備えたプログラミング言語が利用できないため，金額処理を伴ったシステム開発が困難．
(3) 従来のクライアントサーバシステムや端末に比べてレスポンスが悪い．また，企業情報システムが抱えている大量のCOBOL資産（日本の大手企業のCOBOL資産は1社当り10M行～100M行，プログラマの数は全世界で300万人）を利用したいというニーズも強い．

これらの問題を解決するためにCOBOLスクリプト[5),6)]を開発した．本論文では，COBOLスクリプトの開発目標・動作環境・言語仕様・開発環境とインタプリタの実装について述べ，その実用性などを評価する．

2. 開発目標

COBOLスクリプトの最大の開発目的は，現在企業情報システムを開発，保守しているCOBOLプログラマが容易に扱うことのできるWebコンピューティング向けプログラミング言語を提供することにある．このため，COBOLスクリプトは，企業情報システム，特に基幹系システムを，将来Web環境上で再構築する際に必要となる機能を盛り込み実現することを最大の目標とした．これを具体的に述べると次の3点になる．

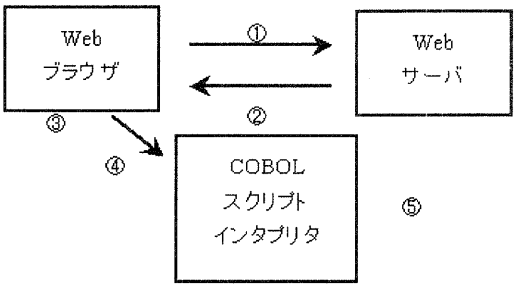
(1) 企業の情報システム部門が現在有している資産（人・プログラム等）やノウハウのスムーズな継承を可能とするためCOBOLの特長を生かした言語仕様とする．
(2) 基幹系システムで実用に耐える性能を確保する．
(3) 大規模開発向けにテスト・デバッグ支援機能をサポートする．

3. 動作環境

COBOLスクリプトは，Webコンピューティングの中核であるWebブラウザ，Webサーバと連携して動作する．ここでは，COBOLスクリプトの動作環境について述べる．

3.1 Webブラウザ上での動作

Webブラウザ上でのCOBOLスクリプトの動作の仕掛けを図1に示す．スクリプトのソースプログラムは，HTMLのSCRIPTタグ中に記述する．Webブラウザはこのタグの記述内容を解析し，インタプリタ



①Webブラウザでホームページアドレスを指定
②Webサーバが指定されたHTMLファイルをWebブラウザに転送
③Webブラウザが転送されたHTMLを解析
④COBOLスクリプトを見つけるとインタプリタを起動
⑤インタプリタがプログラムを解析して実行

図1 COBOL スクリプトの動作
Fig. 1 Action of COBOL script.

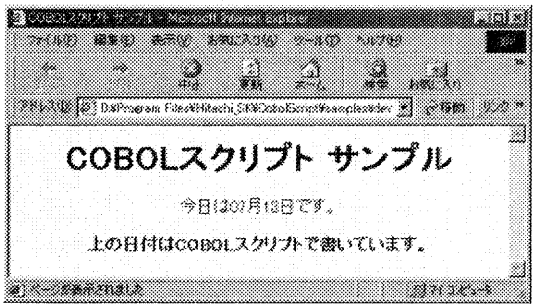


図2 COBOL スクリプトのサンプル（ブラウザ表示例）
Fig. 2 Display sample on browser.

を起動する．スクリプトの例を図2と図3に示す．

3.2 Web サーバ上での動作

Web サーバ上での COBOL スクリプトの動作の仕掛けを図4に示す．

COBOL スクリプトでは CGI コントローラを提供しており，これは，CGI の延長でインタプリタを起動する．CGI コントローラは，URL で指定されたスクリプトファイル（ソースプログラム）を実行するためにインタプリタを起動するだけでなく，CGI アプリケーションで必須となるフォームデータや HTML ファイルをオブジェクトとして操作する環境も提供する．プログラムの一部を例題として図5に示す．

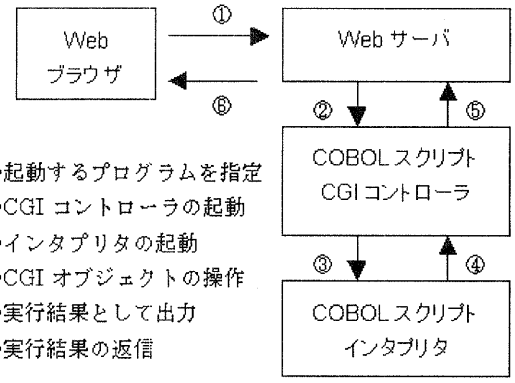
3.3 現在の実装環境

COBOL スクリプトの現在の実装環境を，表1に示す．

この表からも分かるように，COBOL スクリプトのターゲット OS は Windows である．その理由は，次の通りである．

```
<HTML>
<head>
<TITLE>COBOL スクリプト サンプル</TITLE>
</head>
<BODY>
<CENTER>
<H1>COBOL スクリプト サンプル</H1>
<SCRIPT LANGUAGE="COBOLScript">
<!--
IDENTIFICATION DIVISION.
  PROGRAM-ID. サンプル AUTOMATIC PROGRAM.
DATA DIVISION.
WORKING-STORAGE SECTION.
  01 日付時刻   USAGE DISPLAY.
  01 日付表示   USAGE DISPLAY.
  01 ワーク1 USAGE DISPLAY.
  01 ワーク2 USAGE DISPLAY.
  01 日付文字列 USAGE DISPLAY.
PROCEDURE DIVISION.
  MOVE CURRENT-DATE TO 日付時刻
  MOVE CONNECT-STRING(日付時刻(5:2), "月")  TO ワーク1
  MOVE CONNECT-STRING(日付時刻(7:2), "日")  TO ワーク2
  MOVE CONNECT-STRING(ワーク1, ワーク2)  TO 日付文字列
  MOVE CONNECT-STRING("今日は", 日付文字列) TO 日付表示
  MOVE CONNECT-STRING(日付表示, "です。") TO 日付表示
  INVOKE HOST "DOCUMENT.WRITE" USING BY VALUE 日付表示
END PROGRAM.
-->
</SCRIPT>
<P>
<B>上の日付は COBOL スクリプトで書いています。</B>
</CENTER>
</BODY>
</HTML>
```

図3 COBOL スクリプトソースプログラム
Fig. 3 COBOL script source.



①起動するプログラムを指定
②CGI コントローラの起動
③インタプリタの起動
④CGI オブジェクトの操作
⑤実行結果として出力
⑥実行結果の返信

図4 Web サーバ上での COBOL スクリプトの動作
Fig. 4 COBOL script action on Web server.

(1) COBOL の業務は，企業や公共機関を中心に使われており，これら日本のオフィスのデスクトップ環境はほとんど PC である．
(2) サーバ OS としても Windows NT のシェアが拡大してきており，Web サーバという役割であれば Windows NT でも十分こなせる．

また，ブラウザは，現状 Internet Explorer を採用している．これは，Microsoft 社が Internet Explorer とインタプリタとのインタフェースを Active Scripting Interface として公開しているのに対し，Netscape 社の Navigator ではその API を公開していないことによる．Navigator 自身はオープンソースであり，内容

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CGI サンプル AUTOMATIC.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 CGIOBJ OBJECT REFERENCE OLE.
*
01 ワーク.
01 改行.
01 テストタイプ.
01 TEXTPLAIN CONSTANT
     AS "Content-type: text/plain".
PROCEDURE DIVISION.
  INVOKE "CBS.CGIServer.1" "CREATEOBJ"
    RETURNING CGIOBJ
  MOVE CONNECT-STRING(CHAR(14),CHAR(11)) TO 改行
  MOVE CONNECT-STRING(TEXTPLAIN,改行) TO ワーク
  MOVE CONNECT-STRING(ワーク,改行) TO ワーク
  SET "Header" WITH CGIOBJ TO ワーク
  INVOKE  CGIOBJ "GetValue"
   USING VALUE "TESTTYPE" RETURNING テストタイプ
  EVALUATE テストタイプ
    WHEN "CGI 環境変数一覧"
      CALL "CGI 環境変数一覧" USING
        BY CONTENT CGIOBJ
    WHEN "CGI 環境変数値取得"
      CALL "CGI 環境変数値取得" USING
        BY CONTENT CGIOBJ
    WHEN OTHER
      CALL "エラー" USING BY CONTENT CGIOBJ
  END-EVALUATE
  SET CGIOBJ TO NULL
END PROGRAM.
```

図 5 CGI のサンプルプログラム
Fig. 5 Sample program for CGI.

表 1 COBOL スクリプトの実装環境
Table 1 Supported environment for COBOL script.

クライアント OS	Windows 95, Windows 98, Windows NT
Web ブラウザ	Internet Explorer 4.0, 5.0
サーバ OS	Windows NT
Web サーバインタフェース	CGI, ASP[7]

を解析すればインタフェースは分かるが，将来的な互換性を保証する API として定義されていないので，それに対応するインタプリタを作るのが難しい．

4. 言語仕様

4.1 言語仕様に関する検討と設計方針

COBOL スクリプトは以下の観点から COBOL 国際標準規格（いわゆる COBOL85）[8] を参考に言語仕様を検討した．

(1) いわゆるオブジェクト操作のための言語仕様の追加．ブラウザ上で COBOL スクリプトプログラムを実行する場合には，ブラウザ上のテキスト，画像，アプレット等をオブジェクトとして操作できることが必須になる．また，Web サーバ上で COBOL スクリプトを実行する場合には，データベースへのアクセスや，TP モニタを利用したトランザクション処理の実現，実行結果としての HTML 編集・出力が必要であり，これらは将来の拡張性や最新技術動向を考慮すると分散オブジェクトを利用したプログラミングに対応しておく必要がある．
このため，COBOL スクリプトでは，次期 COBOL 規格 [9] で追加予定のオブジェクト指向プログラミング機能の言語仕様の一部（INVOKE, SET など）を先取りし，前に述べた機能を実現することとした．

(2) 現在 COBOL は，銀行，証券，保険など金融システムを始めとして公共，製造業，流通業などの基幹系システムで最もよく利用されている言語である．この最大の理由は，COBOL が長年普及しており，プログラマも多く，標準仕様が存在することにあるが，基幹系で特に使われるのは正確な精度を提供する 10 進演算機能を COBOL が提供していることにある．
C/C++, FORTRAN, Java 等では，算術演算可能なデータ型として固定小数点と浮動小数点をサポートしているが，10 進データはサポートしておらず，特定の形式の文字列と固定小数点・浮動小数点との変換を関数等でサポートしている．シミュレーション等では誤差は問題にならないが，金額計算では誤差は許されない．COBOL スクリプトは，COBOL が最もよく利用されている基幹系システムのユーザを最大のターゲットとしており，10 進演算のサポートは必須である．

(3) コンピュータの 2000 年問題がマスコミでも度々取り沙汰されている．この問題の要因の 1 つに企業情報システムを構成する業務プログラムのライフサイクルの長さが開発者の予想を超えていたことが上げられる．一般的に，業務プログラムの寿命は 10 年をはるかに超えており，この間度々機能追加や仕様変更が行われる．このことから業務プログラムにとってプログラムの可読性は非常に重要な案件であり，国内の企業情報システムの開発・保守に当たっているプログラマにとっては，各社の COBOL がサポートしている日本語プログラミング機能は重要な機能の 1 つである．
このため COBOL スクリプトでは，他のスクリプト言語ではサポートしていないユーザが定義する名前を日本語で表現する日本語プログラミング機能をサポートすることとした．

(4) Web 向けのスクリプト言語は，ソースを直接

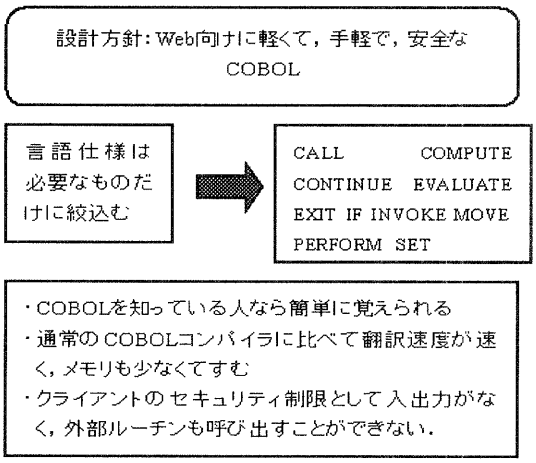


図 6 言語仕様に関する設計方針
Fig. 6 Design policy for language specifications.

実行するインタプリタとして実装されている．特にWebブラウザ上での実行を考えると，プログラムをHTML文書中のテキストの一部として記述する形態が最も簡便であり，実現方式としても一般的である．ソースを直接実行するインタプリタは，プログラムの翻訳から実行までを一度に処理することから，プログラムの実行速度はインタプリタの翻訳処理速度に大きく依存する．

そこでCOBOLスクリプトでは，Webコンピューティングで必要のないCOBOL言語仕様（ファイル処理，GOTO文など）を切り捨てることにより，言語処理系の性能向上を図りやすい仕様とした．この言語仕様の絞り込みに当たり，データを操作するための代入，演算とプログラムを制御するための分岐，サブルーチン呼び出し及び言語の外付けで機能を拡張するためのオブジェクト操作を備えるものとし，処理系が実装する機能の範囲としては，既存のWeb向けスクリプト言語であるMicrosoft社のJavaスクリプト/Visual Basicスクリプトを目標とした．

以上の検討結果から，COBOLスクリプトの言語仕様は，**図6**に示す設計方針で設定した．

4.2 データ型

COBOLスクリプトでは，COBOLプログラマに馴染み易くすることを主目的として，データ型を宣言できるようにした（Javaスクリプト[10),11)]ではデータ型の宣言はできない）．サポートしているデータ型は次の通りである．

(1) 文字項目

文字項目は，USAGE句でDISPLAYと指定した項目で，文字や文字列を格納するデータ項目である．COBOLスクリプトで扱う文字はユニコード（2バイト）なので，1バイト文字（英数字など）と2バイト文字（漢字）の長さを意識しなくてよい．

(2) 数字項目

数字項目は，USAGE DISPLAY句とPICTURE句を指定した項目，またはUSAGE句でBINARY，FLOAT-LONGと指定した項目で，数値を格納するためのデータ項目である．数字項目には，数値がいろいろな表現形式で格納される．数値の表現形式には，外部10進形式，2進形式，浮動小数点形式がある．

外部10進形式 PICUTURE句を指定し，かつ，USAGE句にDISPLAYを指定した項目は，1バイトで数字1けたを表現する．

- 各けたは，0~9の数字でなければならない．
- 演算符号は，数字列の左側に「+」や「−」を置く．
- 数字項目の小数点位置は，PICTURE文字列中の記号「V」で指定する．この小数点は想定小数点として扱われる．
- データ項目中の数字であるべき文字位置の内容が「+」,「−」,「.」，空白等であってはならない．

2進形式 USAGE句でBINARYと指定した項目は，2進形式で表現される．格納できる数値は整数のみで，−2,147,483,648以上，2,147,483,648未満である．

浮動小数点形式 USAGE句でFLOAT-LONGと指定された項目は，倍精度浮動小数点形式で表現される．浮動小数点が表現できる値の範囲及び有効けた数は約16けたで，絶対値で約 2.2×10^{-308}~$1.8 \times 10^{+308}$ 及び0である．この仕様は，次期COBOL規格を参考とした．

(3) オブジェクト参照項目

USAGE句でOBJECT REFERENCE OLEと指定された項目は，オブジェクト参照項目を表現する．スクリプト中からOLEオブジェクトを参照する場合に使用する．この仕様は次期COBOL規格の仕様を拡張した．

(4) バリアント項目

USAGE句でVARIANTと指定された項目は，バリアント項目を表現する．スクリプト中からOLEオブジェクトを参照する場合に使用する．この仕様はCOBOLにないCOBOLスクリプト独自の仕様である．

4.3 文

前述の通り，COBOLスクリプトは標準規格のCOBOLに対して言語仕様をWebで必要とされる

ものに絞り込んだ．ここでは，COBOL スクリプトがサポートしている文とその機能について示す．なお，INVOKE 文，SET 文，EXIT PERFORM 文は，次期 COBOL 規格原案を参考にした．それ以外は，基本的に現行規格のサブセットとした．

(1) **MOVE 文**

MOVE 文は，データを編集・変換して 1 つ以上のデータ領域に移す．一般形式は，次の通り☆．

```
MOVE { 一意名 1 | 定数 1 } TO { 一意名 2 }...
```

定数 1 または一意名 1 のデータ項目は送出し側で，一意名 2 は受取り側である．

(2) **COMPUTE 文**

COMPUTE 文は，算術式の値を 1 つ以上のデータ項目に代入する．一般形式は，次の通り．

```
COMPUTE { 一意名 1 [ROUNDED] } ... = 算術式 1
```

小数点の位置をそろえた後で，算術式 1 の小数部のけた数が一意名 1 のけた数より大きければ，一意名 1 のけた数に従って切捨てが行われる．ROUNDED 指定を書くと切捨て部分の最上位のけたの値が 5 以上の場合，結果の一意名の最下位のけたの絶対値が 1 増やされる（四捨五入）．

(3) **IF 文**

IF 文は，2 岐分岐を記述する．一般形式は，次の通り．

```
IF 条件 1 THEN 文 1 { ELSE 文 2 END-IF | END-IF }
```

IF 文が実行されるとき，次の制御の移行が起こる．

(a) 条件 1 が真の場合，制御は文 1 に移る．文 1 の実行完了後，制御は IF 文の終わりに渡される．

(b) 条件 1 が偽で，ELSE が指定される場合，制御は文 2 に移る．文 2 の実行完了後，制御は IF 文の終わり（END-IF）に渡される．

(c) 条件 1 が偽で，ELSE 指定が書かれていない場合，制御は IF 文の終わり（END-IF）に渡される．

(d) 条件 1 は COBOL と同一仕様であり，ここでは詳細は省略する．

(4) **EVALUATE 文**

EVALUATE 文は，多岐分岐，多岐結合構造（case 構造）を記述する．複数の条件を指定することができ，実行用プログラムの次の動作は，これらの評価の結果に従う．一般形式は，次の通り．

```
EVALUATE 選択主体
{ {WHEN 選択対象 } ... 無条件文 1 } ...
[ WHEN OTHER 無条件文 2 ]
END-EVALUATE

選択主体：
{ 一意名 1 | 定数 1 | 算術式 1 | 条件 1 |
TRUE | FALSE }

選択対象：
{ [NOT] 一意名 2 | [NOT] 定数 2 |
[NOT] 算術式 2 | 条件 2 | TRUE | FALSE }
```

EVALUATE 文の実行は，各選択主体と選択対象が評価され，それらに値，値の範囲，または真理値を割り当て，選択主体と選択対象に割り当てられる値を比較する．比較操作に従って WHEN 指定が選ばれる場合，選択された WHEN 指定の後に続く最初の無条件文 1 を実行する．

どの WHEN 指定も選ばれないで，WHEN OTHER 指定が指定される場合，無条件文 2 を実行する．選択された WHEN 指定の無条件文 1 の終わり，もしくは無条件文 2 の終わりに達するか，または WHEN 指定が選ばれず，WHEN OTHER 指定も指定されない場合，EVALUATE 文の実行は終了する．

(5) **PERFORM 文**

PERFORM 文は，ループを制御する．一般形式は，次の通り．

```
PERFORM { times 指定 | until 指定 | varying 指定 }
無条件文 1
END-PERFORM

times 指定：
{ 一意名 1 | 整数 1 } TIMES

until 指定：
[WITH TEST { AFTER | BEFORE } ] UNTIL 条件 1

varying 指定：
[WITH TEST { BEFORE | AFTER } ] VARYING 一意名 2
```

☆ 以下の表記において，{ | } はその中の要素の 1 つが選択可能なことを示す．また [ ] は要素が省略可能なことを，... はその前の要素が反復可能なことを示す．

```
FROM { 一意名3 | 定数1 } BY { 一意名4 | 定数2 }
UNTIL 条件1
```

(a) TIMES指定は，整数1または一意名1によって指定される回数だけ無条件文1を実行する．
(b) UNTIL指定は，条件1が真になるまで無条件文1を実行する．
(c) VARYING指定は，一意名3または定数1の値で一意名2を初期化した後，条件1が真になるまで無条件文1を実行すると共に，一意名2に一意名4または定数2の値を加算する．

(6) **EXIT文**

EXIT文は，一連の手続きに共通の出口を提供する．EXIT PROGRAM文は，プログラムの論理的な終わりを示し，EXIT PERFORM文は，PERFORM文を終了させる手段を提供する．一般形式は，次の通り．

書き方1

```
EXIT PROGRAM
```

書き方2

```
EXIT PERFORM [CYCLE]
```

CYCLE指定のないEXIT PERFORM文の実行は，制御を次のEND-PERFORM指定の直後に渡す．CYCLE指定のあるEXIT PERFORM文の実行は，制御を次のEND-PERFORM指定の直前にある暗黙のCONTINUE文に渡す（ループの先頭に戻す）．構造化プログラミングをするときはGOTO文は不要であるが，ループからの脱出用の分岐文だけは欲しくなる．このため，次期COBOL規格に入るEXIT PERFORM文をサポートした．

(7) **CALL文**

CALL文は，指定したプログラムに制御を移す．一般形式は，次の通り．

```
CALL 定数1
[ USING { [BY REFERENCE] { 一意名2 } ...
| BY CONTENT { 一意名2 } ... } ... ]
```

CALL文のUSING指定中の引数と，呼ばれるプログラムの手続き部見出しのUSING指定中の引数は，出現順序による対応であり，名前の等価性にはよらない．一意名2がオブジェクトデータの場合，BY CONTENT指定を使用する．

(8) **INVOKE文**

INVOKE文は，オブジェクトのインスタンスの生成やメソッド呼び出しを行うことができる．一般形式は，次の通り．

書き方1

```
INVOKE { 一意名1 | 定数1 } { 一意名2 | 定数2 }
[USING {BY VALUE {
{ 一意名3 | 定数3 } [NAMED { 一意名4 | 定数4 }]
| NO DATA
} ... } ... ]
[RETURNING 一意名5]
```

書き方2

```
INVOKE { 一意名6 | 定数5 } "CREATEOBJ"
RETURNING 一意名7
```

書き方3

```
INVOKE { 一意名8 | 定数6 } "GETOBJ"
[USING BY VALUE { 一意名9 | 定数7 }]
RETURNING 一意名10
```

INVOKE文は，COBOLスクリプト処理系の外部にあるオブジェクトを操作する機構を提供する．一般的に，GUIの処理記述やデータベース操作及び既存のCOBOL処理系で作成したロードモジュールの呼び出し等に用いる．

(9) **SET文**

SET文は，オブジェクトの属性値の取得や設定を行う．一般形式は，次の通り．

書き方1

```
SET 一意名1 TO { 一意名2 | 定数1 }
WITH { 一意名3 | 定数2 }
```

書き方2

```
SET { 一意名2 | 定数1 }
WITH { 一意名3 | 定数2 } TO { 一意名1 | 定数3 }
```

書き方3

```
SET 一意名4 TO { NULL | EMPTY }
```

書き方4

```
SET 一意名5 TO 一意名6
```

書き方1は属性値を取得し，書き方2は属性値を設定する．書き方3はオブジェクト参照項目に格納され

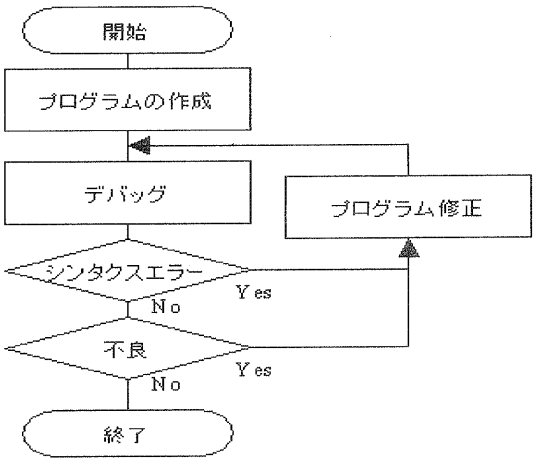


図7 ローカルマシン上のテストデバッグ手順
Fig. 7 Test debug process on local machine.

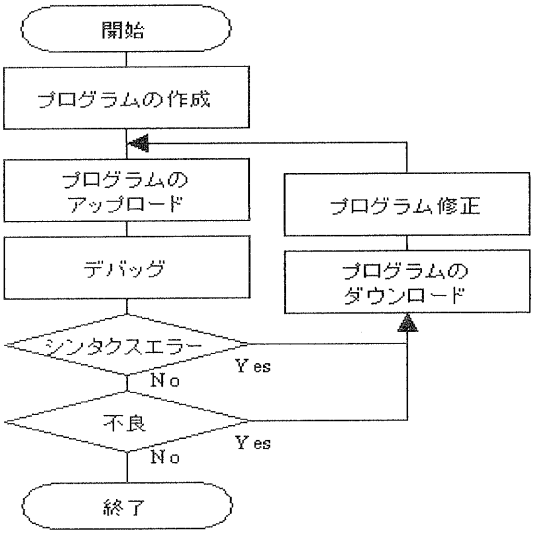


図8 Webサーバ上のテストデバッグ手順
Fig. 8 Test debug process on Web server.

ているインスタンスの解放および NULL，EMPTY 値の設定に用いる．SET 文は，INVOKE 文と同様 COBOL スクリプト処理系の外部にあるオブジェクトを操作する機構を提供するものであり，GUI の処理記述やデータベース操作に用いる．

(10) **CONTINUE 文**

CONTINUE 文は，無操作文であり実行文が存在していないことを示す．CONTINUE 文は，条件文または無条件文が使用可能な場所ならどこでも使用することができる．

5. 開発環境

5.1 デバッガ

COBOL スクリプトデバッガは，プログラムのソースコードを画面上に表示し，確認しながらテストデバッグできるツールである．COBOL スクリプトデバッガには次の機能がある．

(1) 被テストプログラムのソースコード表示
(2) スクリプト（プログラム）の実行制御
(3) データ名（変数）の値の表示と代入
(4) サーバプログラムのリモートデバッグ

完成した COBOL スクリプトのプログラムは通常サーバに配置するが，デバッグはプログラマ自身の PC（ローカルマシン）で行うことができる．ローカルマシン上でのテストデバッグの流れを図7に示す．

(1) COBOL スクリプトを使用した HTML ファイルやスクリプトファイルをローカルマシン上に作成する．
(2) COBOL スクリプトデバッガを使用して作成したプログラムをブラウザに読み込ませデバッグを開始する．
(3) シンタックスエラーやプログラム不良が存在する間は，プログラムを修正する．
  (a) シンタックスエラーの場合
  エディタを使用してシンタックスエラーが存在する HTML ファイルを開き，メッセージウィンドウに表示されている行番号へカーソルを位置づけ，エラー箇所を修正する．
  (b) プログラム不良の場合
  エディタを使用してプログラム不良が存在する HTML ファイルを開き，不良箇所を修正する．
(4) プログラム不良が存在しなくなった時点でテスト完了．

次に Web サーバでのテストデバッグの流れを図8に示す．

(1) COBOL スクリプト言語を使用して HTML ファイルやスクリプトファイルをローカルマシン上に作成する．
(2) Web サーバに作成したプログラムをアップロードする（ftp などを使用して行う）．
(3) COBOL スクリプトデバッガを使用して作成したプログラムをブラウザに読み込ませデバッグを開始する．
(4) シンタックスエラーやプログラム不良が存在する間，プログラムをサーバ上からクライアントのローカルにダウンロードし，プログラムを修

表2 C1メジャーの対象となる文
Table 2 Target statements for C1 measure.

文	分岐数
IF	2
EVALUATE	WHEN OTHERが指定されている場合は，WHENの数 WHEN OTHERが指定されていない場合は，WHENの数+1
PERFORM	2

正する．

(a) シンタックスエラーの場合
エディタを使用してシンタックスエラーが存在するHTMLファイルを開き，メッセージウィンドウに表示されている行番号へカーソルを位置づけ，エラー箇所を修正する．

(b) プログラム不良の場合
エディタを使用してプログラム不良が存在するHTMLファイルを開き，不良箇所を修正する．修正したプログラムをWebサーバ上に再アップロードし，再度デバッグを行う．

(5) プログラム不良が存在しなくなった時点でテスト完了．

5.2 カバレージ機能

カバレージ機能は，プログラムごとのテストの進捗状況を定量的に管理するための機能であり，基幹系システムの品質を保証するために，キーとなる機能であり，COBOLスクリプトがサポートしている他のスクリプト言語にない特長の1つである．COBOLスクリプトでは，C1メジャーのカバレージをサポートしている．C1メジャーとは，全体の分岐の数に対する実行済みの分岐の数の割合を示す情報であり次の式で示される．

C1メジャー＝実行済みの分岐の数／実行可能な分岐の総数 × 100（%）

C1メジャーの対象となる文を**表2**に示す．

COBOLスクリプトでは，カバレージ情報をCSV形式のテキストファイルに，次に示すフォーマットで，それぞれを改行して出力する．

(1) COBOLスクリプトカバレージ情報のタイトル
(2) カバレージ対象のプログラムが含まれるファイル名
(3) ファイル最終更新日時
(4) プログラム名
(5) カバレージ取得日時
(6) カバレージ取得回数
(7) C1カバレージ対象文総数
(8) C1カバレージ実行済文数
(9) C1カバレージ未実行文数
(10) C1カバレージ率
(11) 行ごとに行番号と実行／不実行表示

(1)～(10)までは必ず出力する．(11)以降がない場合は，C1取得文を使用しない場合でこの場合(7)～(10)までの値は0となる．

6. インタプリタの実装

6.1 性　　能

COBOLスクリプトは，Web環境のインタプリタで実行される．インタプリタの実装に当たって，特に性能の観点から次の3点を考慮した．

(1) 翻訳処理の高速化
(2) 実行処理の高速化
(3) シンクライアント☆での使用を想定したメモリ所要量の削減

COBOLスクリプトは，言語仕様上は標準規格COBOLに類似しているが，字句解析及び構文解析の観点から見ると標準規格COBOLに比べて非常に単純化されている．このため字句解析及び構文解析処理はCOBOLスクリプト専用に最適化して設計すれば，既存のCOBOL処理系に比べて高速かつ軽量に作成できる．

例えば，この実例としてCOBOLのAUTHOR段落がある．AUTHOR段落は，プログラムの作成者の名前を記述するためのものであり，予約語を含むどのような文字列を記述してもよい．このため，COBOLコンパイラの字句解析処理は，構文解析処理と何らかのインタフェースを持つか，部分的な構文解析処理を実装しなければならない．実際，COBOLスクリプトの製品化に当たっては，既存のCOBOL処理系の流用は行わず新規に設計しなおした．これにより，翻訳処理は市販されている既存のCOBOLコンパイラである日立のCOBOL85及びマイクロフォーカス社のCOBOL 3.1Jと比較して約1.5～2倍高速という結果を得ている．

逆に，実行処理に関しては，既存のCOBOL処理

☆ ネットワークでの使用を前提とした軽量クライアント．処理のほとんどをサーバ側で行うためCPUやメモリなど十分でない場合が多い．

表 3 性能比較（単位 秒）
Table 3 Performance comparison (second).

	COBOL スクリプト	Java スクリプト
固定小数点	10	16
浮動小数点	10	16
10 進	62	82

[測定条件]
- 演算，比較，転記の組み合せ合計 27 ステートメントを 10 万回ループ
- CPU： Pentium II 266MHz，メモリ 64MB

系が他言語に比べて得意な部分を活用することにより高速化を図っている．一般的に，COBOL 処理系と C 言語処理系を比較すると，文字列操作や 10 進演算で COBOL が速く，固定小数点演算や浮動小数点演算では C が速い．これは，言語として活用される場が異なるために，言語処理系としての最適化のポイントが異なるためである．特に COBOL が得意な部分に関しては，専用のデータ型を言語として定めていることが最適化を可能にしている．

COBOL スクリプトも COBOL のデータ型をサポートしているため，高速なインタプリタの設計が可能となっている．特に，製品版の実装に当たっては，10 進演算のアルゴリズム [12] はアセンブラで記述し高速化を図っている．ちなみに，10 進演算以外は C++を用いて記述しているが，これはブラウザインタフェースやオブジェクト操作のためのインタフェースが C++ のクラスライブラリとして提供されていることによる．

COBOL スクリプトと反対の典型的なスクリプト言語が，言語仕様上データ型の宣言を持たない Java スクリプトである．実際に製品で比較すると，文字列操作，固定小数点演算，浮動小数点演算，10 進演算のいずれにおいても COBOL スクリプトが 1.2～1.6 倍高速という結果を得ている．なお，Java スクリプトでは 10 進演算は不可能なため，数字文字列を数値に変換し演算した結果を数字文字列に変換しなおすプログラムで代替している．このため，演算自体は，浮動小数点を用いたものとなっており，厳密には 10 進演算としての精度保証はなされていないため，あくまで参考のために示したものである．性能測定結果の一例を**表 3** に示す．

最後に，メモリ所要量であるが，翻訳性能の高速化同様，言語仕様の単純化が大きく寄与しており，既存の COBOL コンパイラに比べて 1/3 程度で実現できている．なお，データ型を持たない Java スクリプトに比べても 20%程度の増加に止まっており，実用上問題ない範囲となっている．

6.2 Web ブラウザと Web サーバでの実装上の相違

COBOL スクリプトの動作環境としては，Web ブラウザと Web サーバがあることを 3 章で述べたが，インタプリタの実装の点でこれらに相違がある．これは，次の理由による．

(1) COBOL スクリプトで記述したアプリケーションプログラムを Web ブラウザ上で実行させる場合，セキュリティの点で実行環境のコンピュータのローカルなリソースにアプリケーションプログラムがアクセスする機構を提供すべきでない．

(2) COBOL スクリプトで記述したアプリケーションプログラムを Web サーバ上で実行させる場合，アプリケーションプログラムはサーバ上のリソース（例えばデータベース）にアクセスしてエンドユーザに対するサービスを提供する．

プログラムを実行するコンピュータのリソースにアクセスする仕掛けは，従来の COBOL 処理系では，入出力機能や通信機能として言語仕様に組み込まれており，これらを実行する文として OPEN, READ, WRITE, CLOSE 文などがある．これに対して COBOL スクリプトでは，ファイル処理を行うためのこれらの文をサポートしていない．また，一般的な COBOL 処理系では，CALL 文を用いて C 言語やアセンブラで記述したプログラムを呼び出せるようになっており，これを利用して OS やミドルソフトが提供する API を直接利用することできるが，COBOL スクリプトでは CALL 文は呼び出し元と同じ HTML 文書に存在する COBOL スクリプトのプログラムだけを呼び出すことができる．

一方，サーバ上のアプリケーションプログラムがデータベース等にアクセスするための仕掛けとして，オブジェクトを操作するための INVOKE 文，SET 文がある．これにより，例えばマイクロソフト社が提供する ADO（Active Data Object）等のデータアクセスオブジェクトを利用してデータベースをアクセスできる．なお，ブラウザ上ではこれらの文を記述することは可能であるが，インスタンスを新たに生成することを禁止しており，ブラウザが HTML 文書から生成したインスタンス（例えば Java アプレット）にのみアクセスできる．

Web サーバ上と Web ブラウザ上における COBOL スクリプトの実装上の相違をまとめると次のようになる．

(1) シンタクス等の純然たる言語仕様に相違はない．

(2) Web ブラウザ及び Web サーバとインタプリタ

を接続するインタフェース部分は異なる．

(3) Webブラウザ上ではINVOKE文を用いてインスタンス生成を行う機構を提供していない．

6.3 通信機能の実装

COBOLスクリプトでは，Javaスクリプトなどの他のスクリプト言語にはない特長として，httpプロトコルを利用した通信機能を組み込みオブジェクトとして実装している．これは，既存の業務プログラムでよく利用されている単票形式のフォームを利用したデータエントリやデータ表示を，HTMLで記述した画面の再読み込みなしにデータの送受信のみで実現することを目的としたものである．このオブジェクトは，次のメソッド及びプロパティを実装している．

(1) putメソッド
指定された引数を，送信用のバッファに出力する．

(2) sendRequestメソッド
引数で指定されたサーバにhttpプロトコルでデータを送信する．

(3) readメソッド
サーバからの返信データを受け取る．

(4) userID
ユーザIDを格納するプロパティ．

(5) password
パスワードを格納するプロパティ．

(6) receiveTimeOut
データのインターネット受信要求で使用されるミリ秒単位（1/1000秒）のタイムアウト値．

7. 評価と今後の課題

COBOLスクリプトの実用上の評価と今後の課題は，次の通りである．

(1) 現在COBOLスクリプトを評価中の企業は100社を超えており，実用的な運用に向けた検討も進んできている．情報システム部門を中心としたユーザ約100名に対するアンケートの結果，半数以上のユーザから実際に活用できるという評価を得た．特に「COBOLでここまでできるのか」という驚きを伴った評価も多く，COBOLプログラマ向けWebコンピューティング向けプログラミング言語として一定の評価が得られたと判断する．

(2) 実装面では，他のスクリプト言語と比べて性能上同等以上という結果が得られており，現在のWebの利用環境において実用的なレベルに達していると判断できる．ただし，今後の膨大な利用者を背景としたインターネットビジネス分野での利用については，更に評価を加える必要がある．

(3) COBOL既存資産（プログラム）を流用したWebシステム開発については，潜在的なニーズは非常に高いが，現状では2000年問題への対応が優先課題となっており，実際に運用を始めている例はない．このため，実際の評価は難しいが，COBOLスクリプトはCOBOLの言語仕様を単純化しており，単純に既存資産を流用することは困難と推測している．言語以外のインフラの点でも既存システムからWebシステムへの単純移行は不可能であるが，移行コストを下げるための手段を現在検討中である．

(4) 本論では特に触れていないが，セキュリティの面では他のスクリプト言語と同様に一般的なSSLを利用することが出来る．これは，クレジットカード番号等の個人データの暗号化などで有効である．しかし，企業情報システムのプログラム自体が一般的に企業秘密であることを考えると，ソースプログラムを単純にダウンロードして実行する現在の仕掛けは問題ありと考える．SSLを用いていないページにおいてもソースコードを隠匿する何らかの手段が必要であり，現在検討中である．

8. まとめ

COBOLプログラマが容易に扱うことができるWebコンピューティング向けプログラミング言語COBOLスクリプトについて述べてきた．COBOLスクリプトでは，他のスクリプト言語にない次の機能を実現できた．

(1) COBOLの特徴を継承し，10進形式のデータ型をサポートした．この結果，金額計算など誤差の許されない分野の業務をWeb上で心配なく稼働可能とした．

(2) 大規模な開発プロジェクトでの利用に備えて，定量的なテスト進捗管理を支援するカバレージ機能をサポートした．

(3) プログラムの保守性を向上する日本語プログラミング機能をサポートした．

(4) 他のスクリプト言語に比べて優れた実行性能を実現できた．

マーケティングの観点からは，COBOLスクリプトには次の課題がある．これらは，技術的なテーマではないが，いずれも「商品」としては重要な事項である．

(1) 稼働環境：Microsoft社Internet Explorerで

しか稼働しない．ブラウザとスクリプト言語のインタフェース API が公開されていないので Netscape Navigator 環境がサポートされていない．

(2) 海外での普及：日本語版だけで，海外版がサポートされていない．

(3) 標準化：JavaScript は ECMAScript として ECMA および ISO で標準化されたが，COBOL スクリプトを含め他のスクリプト言語は標準化されていない．

(4) 健全な市場形成：Java スクリプトや Visual Basic スクリプトは他製品の普及のために無償であるが，COBOL スクリプトは単独の製品で，有償である．無償製品（おまけ）が主流だと，本分野（スクリプト言語）の健全な進歩と競争が損なわれる恐れがある．

参 考 文 献

1) World Wide Web Consortium: HTML 4.0 Specification, http://www.w3.org/TR/1998/REC-html40-19980424/html40.txt (1998).
2) Gosling, J., Joy, B. and Steele, G.: The Java Language Specification, Addison-Wesley Publishing Company (1996). http://java.sun.com/docs/books/jls/html/index.html.
3) NCSA HTTPd Development Team: The CGI Specification, http://hoohoo.ncsa.uiuc.edu/cgi/interface.html.
4) Microsoft Corporation: VBScript Language Reference, http://msdn.microsoft.com/scripting/vbscript/doc/vbstoc.htm (1999).
5) 日立ソフトウェアエンジニアリング（株）：COBOL スクリプト, http://www.np.hitachi-sk.co.jp/cobolscript/ (1998).
6) （株）日立製作所：COBOL スクリプトユーザーズガイド (1998).
7) Microsoft Corporation: Active Server Pages, http://msdn.microsoft.com/library/sdkdoc/iisref/iiwawelc.htm (1999).
8) JIS X 3002-1992 電子計算機プログラム言語 COBOL, 日本規格協会 (1992).
9) Committee Draft 1.5 Proposed Revision of ISO 1989: 1985 Programming language COBOL (1999).
10) Netscape Communications Corporation: Client-Side JavaScript Reference v1.3, http://developer.netscape.com/docs/manuals/js/client/jsref/index.htm (1999).
11) ECMA: ECMAScript Language Specification, http://www.ecma.ch/stand/ecma-262.htm (1998).
12) Knuth, D. E.: Seminumerical Algorithms, The Art of Computer Programming Volume II, Addison-Wesley, 2nd edition (1981).（中川 圭介訳：準数値算法/算術演算, サイエンス社 (1986).）

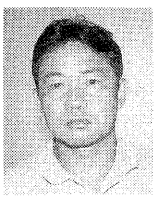
(平成 11 年 7 月 16 日受付)
(平成 11 年 12 月 29 日採録)

三宅 立記（正会員）

昭和 35 年生．昭和 57 年九州大学理学部物理学科卒業．同年，日立ソフトウェアエンジニアリング（株）入社．COBOL コンパイラ，Web 関連スクリプト言語のインタプリタの開発に従事．

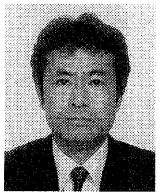
今城 哲二（正会員）

昭和 21 年生．昭和 44 年国際基督教大学教養学部自然科学科（数学専攻）卒業．同年，（株）日立製作所入社．COBOL などのコンパイラ，第 4 世代言語，ソフトウェア開発環境（CASD, EAGLE, SEWB）の開発に従事．現在は，ソフトェア事業部マーケティング推進部長．長年，CODASYL COBOL 委員会委員，JIS COBOL 原案作成委員会委員長など標準化委員を歴任．1999 年，情報処理学会情報規格調査会標準化貢献賞受賞．1995 年より，千葉大学工学部非常勤講師．1997 年より，奈良先端科学技術大学院大学博士課程在学中．プログラム言語，コンパイラ，ソフトウェア工学，日本語処理と国際化に関心をもつ．著書「明解 COBOL」（サイエンス社），「ビジネスオブジェクト入門（編著）」（SRC），「標準 COBOL プログラミング（編著）」（カットシステム）．電子情報通信学会，情報処理学会，日本ソフトウェア科学会会員．



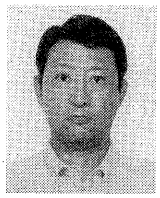
佐藤 忍（正会員）

昭和 32 年生．昭和 57 年東京大学理学系研究科数学専攻課程修了．同年日立ソフトウェアエンジニアリング（株）入社．プログラミング言語のコンパイラ開発に従事．



井藤 敏之（正会員）

昭和27年生．昭和49年神戸大学工学部電子工学科卒業．同年，日立ソフトウェアエンジニアリング（株）入社．画面エディタの開発，ユティリティの開発，COBOLコンパイラの開発，開発支援ツールの開発に従事．現在同社九州開発センタ長．



横塚 大典（正会員）

昭和35年生．昭和59年京都大学法学部卒業．同年（株）日立製作所入社．平成2年 社費留学にてエジンバラ大コンピュータサイエンス学科入学．平成3年 同修士課程修了．現在はソフトウェア事業部にてCOBOLコンパイラ，オブジェクト指向COBOLコンパイラの開発に従事．

辻畑 好秀（正会員）

昭和25年生．昭和48年東京大学基礎科学科卒業．同年（株）日立製作所入社．ソフトウェア事業部にてプログラミング言語のコンパイラ開発に従事．

植村 俊亮（正会員）

昭和41年京都大学大学院工学研究科修士課程修了．同年電子技術総合研究所（当時名称電気試験所）入所．昭和63年東京農工大学教授（工学部数理情報工学科）．平成5年奈良先端科学技術大学院大学情報科学研究科教授．工学博士．データベースシステム，自然言語処理，プログラム言語の研究に従事．情報処理学会，ACM，IEEEなどの会員．