LO-1

プログラム資産活用のための再構造化

Restructuring of Legacy Programs for Reuse

梶尾義規† 魚田勝臣‡ 永田守男†
Yoshinori KAJIO Katsuomi UOTA Morio NAGATA

1. はじめに

企業が旧来から保持しているCOBOL資産をもとに新しい技術によるプロダクトへ再構築する方途が多数提案されている．これらは元となるCOBOLプログラムからデータの定義や手続きの流れなどの情報を抽出し，新しい言語によるプログラムやインタフェースを生成する．しかし，既存のプログラム資産を進化させて活用する視点に欠け，新しい開発技術を導入している．

われわれは，COBOL資産を新しい技術や状況に応じて進化させやすく，過去に蓄積した開発技術を継承できる情報システムの再構築法の確立を目指している．再構築の過程で，(1)既存のCOBOLプログラムを新しい技術のソフトウェアに適した構造に変換し，(2)変換された結果から新しい技術によるプロダクトを生成する．これを実現するために，中間にオブジェクト指向COBOL（OO-COBOL）を導入して[1,2]，新しい技術による情報システムの構築にこれまでのCOBOL資産を活用する．

本稿では，これらの方法のうち，主に既存のCOBOLプログラムを元にした階層構造を持つOO-COBOLプログラムの生成法について述べる．

2. COBOL資産再利用の現状

現在COBOL資産を再利用する技術や方法の提案として次のようなものがある．

(1) 新技術への移行の取り組み

COBOLプログラムを再利用して新しい技術によるプロダクトを生成するために，自動的あるいは半自動的な変換技法が企業などにより提案されている[3,4]．その仕組みは次の3つに分けられる．

a. COBOLプログラムを入力として，CORBAやCOMなどの分散環境からそのCOBOLプログラムを呼び出すためのコンポーネントを生成するもの．
b. COBOLプログラムを入力として，JavaやSQLなど他の言語のプログラムに変換するもの．
c. 古い規格のCOBOLプログラムをより新しい規格のCOBOLや特定ベンダのCOBOLシステムに対応させるよう変換するもの．

プログラム言語の変わらないc.を除いて，これらの方法は既存のCOBOL資産をそのままにして外部の新しい技術と連携させるか，または新しい技術に置き換える．a.の場合も，システムの進化にCOBOL資産が十分に活用されず，過去に蓄積した開発技術の継承が困難になるという問題点がある．

†慶應義塾大学大学院 理工学研究科
‡専修大学大学院 経営学研究科

(2) オブジェクト指向による再構成の取り組み

COBOLプログラムをオブジェクト指向の設計に活用する方法も提案されている[5]．これは，一連のプログラムをもとに同等の機能を実現するオブジェクト指向の設計情報を生成する．まず a.制御フロー分析とデータフロー分析を行ってCOBOLのデータや手続きをオブジェクト指向のクラスの属性や操作に割り当て，次いで b.小さなクラスの除去や冗長な操作の分解などを行い設計の改善を図る．

しかし，元からあったデータや手続きの構造がほとんど失われてしまうため，新しい設計情報を把握することが困難である．

こうした技術の提案が盛んなのは，情報システムへの要請が集中型処理から分散型処理に移行する中で，過去にCOBOLで構築した業務ロジックを再利用する方途が求められているからである．この問題に対し，膨大なCOBOL資産とその開発技術を活用するためには，新しい技術と連携させたり置き換えるだけではなく，既存のCOBOLプログラムの構造を再利用しやすいものに変えて，COBOL資産の長期的な進化を図る考え方が必要である．

3. OO-COBOLを介在させた再利用の提案

われわれは，既存のCOBOLプログラムを一度OO-COBOLプログラムに変換して構造も分散型処理に適したものにした上で，他の技術によるプロダクトに変換，接続する方法を提案する．

(1) 対象とする処理

再利用の対象は，COBOLのプログラムでデータの追加，更新，削除などのトランザクション処理を行っているものである．このプログラムを，半自動的な方法でOO-COBOLによるトランザクション処理に変換する(図1)．一連のプログラムの構造も，手続き指向からオブジェクト指向に変える．

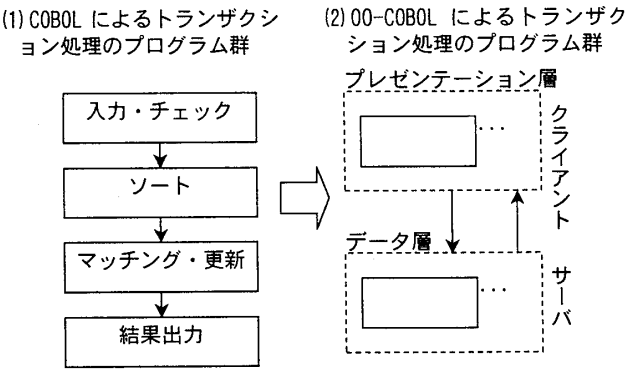

図1 COBOL プログラムと OO-COBOL プログラムとの対応づけ

COBOLプログラムは組織内で一定のコーディング規則を定め，処理の流れの組み方，データ項目の命名法などを統一していることが多い．元となる COBOLプログラムが一定のコーディング規則に従っていることを前提として，それを手がかりに半自動的にOO-COBOLプログラムに変換する．

OO-COBOLは，従来のCOBOLの文法と機能をサブセットとして含み，クラスや継承などオブジェクト指向の要素を追加している．新しいプログラム言語などの技術を導入する場合に比べ，新たに必要となる開発技術が少なくて済むようになる．

(2) COBOL から OO-COBOL へのプログラムの抽出･変換

COBOLのトランザクションからOO-COBOLのトランザクションにプログラムを変換する際には，元となるCOBOLプログラムの構文解析を行いながら特定の要素を抽出し，必要な変換を施す．要素の抽出と変換の方法は，既存のCOBOLプログラムを元に同等の機能を持ち階層構造からなるOO-COBOLプログラムを人手で作り，両者を比較して考案した．

変換の方法は，元のCOBOLプログラムのロジック，また変換して作りたいOO-COBOLプログラムのロジックによって変わる．抽出の方法は，複雑さの点から次の3つに分かれる．

a. 簡単なサーチと，特定の要素の名前(ファイル名等)を指定するだけで導出できるもの．
b. a.の方法をもとに抽出する候補をある程度絞り込み，人手によって決定できるもの．
c. 元のCOBOLプログラムのロジックに依存して方法が変わるもの．

2で述べたようなプログラムの構造を大きく変える方法は，新しいプログラムの把握に必要な労力が大きい．元になるものとそれから新たに生成するものの構造や処理の流れがある程度定まっている変換法は，生成されるプログラムの内容を把握しやすく，構造の改善と必要な労力の低減の双方を満たす．

(3) プログラム変換ソフトウェア

(2)の仕組みによりCOBOLからOO-COBOLへの変換を行うソフトウェアをJavaで実装した．その処理の流れは，a.COBOLプログラムを構文解析してプログラムの構造を分析するフェーズ，b.変換すべき要素を必要に応じて自動，半自動，手動で抽出して変換を施し，作りたいOO-COBOLプログラムの構造を生成するフェーズ，c.生成されたOO-COBOLプログラムの構造を元に実際のOO-COBOL原始プログラムを出力するフェーズ，の3つに分かれる．

(4) OO-COBOL からの他の技術によるプロダクトの構築

COBOLプログラムをOO-COBOLプログラムに変換し，構造もクライアントとサーバに分けた上で，これを他のソフトウェア技術によるプロダクトの構築に用いる(図2)．OO-COBOLもCOBOLと同様に他言語との連携機能が確立されており，OO-COBOLプログラムを外部とのインタフェースの実装に役立てられる．またユーザ･インタフェース(UI)のロジックをWebのUIの構築に用いたり，データの定義をXMLによるデータの定義に変換するなどの活用も可能である．

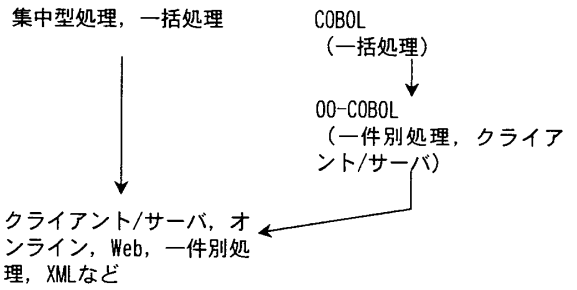

図2 OO-COBOLを介在させた技術の移行

一度オブジェクト指向の構造にしてから他の技術に移行することで，次の利点が得られる．

a. 既存のCOBOL資産がその後も進化しやすい構造になる．
b. 蓄積した資産からの変化の経緯を容易に把握でき，新しい開発技術の導入が行いやすくなる．

以上の提案のうち，これまでに(3)のソフトウェア開発までを行い，COBOLプログラムを人手による判断を受けながらOO-COBOLプログラムに変換できることを確かめた．

OO-COBOLは既存のCOBOLプログラムと新しい技術とのインタフェースの構築などに使用されてはいるものの，その他の利用の方途は普及していない．膨大なCOBOL資産を今後も進化させ，新しい環境に適応しやすくするために，OO-COBOLを活用し，既存のCOBOLプログラムの構造の改善と新しいプロダクトの構築を行うべきである．

4. おわりに

COBOL資産の再利用とOO-COBOLの活用に新たな視点を与え，現実を見据えた再構築，システムの長期的な進化，蓄積された技術の継承などの点で優れる方法を提案した．

本稿で述べたことは，企業で実際に使用されているCOBOLプログラムを元に方法･ソフトウェアの開発を行った結果，得ることができた．他のコーディング規則に基づく場合にもあてはめ，適用範囲を確かめることが今後の課題である．

参考文献

[1] Doke, E. R., Hardgrave, B. C.: An introduction to object COBOL, Wiley & Sons (1998).
[2] Chapin, N.: Standard Object-Oriented COBOL, Wiley & Sons (1997).
[3] Micro Focus: Legacy transformation and integration, http://www.microfocus.com/files/whitepapers/elinkwhitepaper3.pdf
[4] NEC: Open COBOL Factory 21 / Object Partner Pro, http://www.sw.nec.co.jp/cced/ocf21/objptnpro/seihin.html
[5] Joiner, J. K., Tsai, W. T.: Re-engineering legacy Cobol programs, CACM, Vol. 41, No. 5, pp. 185-197 (1998).