業務システムのモダナイゼーション支援技術

An Approach to Transforming Systems for Execution on Cloud Computing System

● 前田芳晴　● 上村　学　● 矢野啓介

あらまし

ビジネスのデジタル化が加速する中，どのようにして既存の業務システムをICTやビジネスの変化に対応させていくかが課題となっている。システムを仮想化してクラウドに移行するだけでなく，業務拡大や他サービスとの連携など，ソフトウェアを柔軟性の高い構造に移行することが望まれている。一方，稼働中のシステム全体を新たに作り直すのは，コストやリスクの観点から現実的ではない。

本稿では，システム全体の移行ではなく，システムから部分を抽出し，部分ごとに特性に合わせた形に移行していくアプローチを示し，その実現を支援する三つの技術を紹介する。まず，ソフトウェア地図を作成してプログラム資産の全体の概観と機能間の関係を可視化する。次に，業務ロジック複雑度や更新頻度などのプログラムの特徴量を算出して地図上の建物の高さに設定することによって，移行対象の機能とプログラムの優先付けを行う。最後に，優先付けしたプログラムをシンボリック実行によって解析して，実装された業務上の決まりや計算の方法などを理解しやすい条件表の形式で抽出する。本アプローチによって，SoR(Systems of Record)である既存システムから機能部分を抽出し，機能ごとの移行の方法として，例えばAPI(Application Programming Interface)の切り出しによるサービス化，BRMS(Business Rules Management System)化，あるいは継続活用のような計画立案が可能となる。

Abstract

Business digitalization continues to accelerate, and adapting existing enterprise systems to changing business practices and advances in information and communications technology (ICT) is a growing problem. Not only must current systems be virtualized for execution on a cloud computing system but also software must be restructured with high flexibility to meet expanding business requirements, such as coordinating with other services. However, it is usually not feasible to re-implement an entire system due to the cost and the risk of system malfunction. In this paper, we present an approach to transforming a system so that it has more flexibility and expandability. It works by extracting each part of the system, analyzing its characteristics, and identifying an appropriate implementation for it in accordance with its characteristics. First, the structure of the system and the relationships among functions are visualized by analyzing the program files. Next, the business logic complexity, update frequency, etc. of each program is characterized. The feature values obtained are assigned to the heights of the corresponding structures on the software map and are used to characterize and prioritize the function or the program. Finally, the prioritized program is analyzed by using symbolic execution, and the rules or calculation methods used in the business are extracted as a decision table in a comprehensible format. This approach enables an existing system, the "system of record (SoR)," to be transformed by extracting its features and identifying the best solution for each feature such as defining it as a service, using it with a business rules management system (BRMS), or using the program as is.

ま え が き

クラウドやモバイルなどのICTが発展していく中，業務システムの課題の一つは，ビジネス環境やICTの変化に対応して時代に合わせた望ましい形に移行していくことである。

業務システムの移行方法の一つであるシステム仮想化によるIaaS（Infrastructure as a Service）への移行は，ホストコンピュータなどのハードウェアの運用費用などを削減できる。しかし，システムのプログラムやアーキテクチャーは従来どおりであり変更容易性などは改善されないため，効果が限定的になると考えられる。移行後のシステムが，種々の変化に対して長期にわたって高い柔軟性を備えるためには，ハードウェアだけでなくプログラムなどのソフトウェアも継続的にモダナイズすることが必要である。

既存ソフトウェアのモダナイゼーションには，新規開発にはない難しさがある。それは，

(1) 長期間の部分的な改修や変則的な変更が重畳されて，プログラムが複雑化していること
(2) 改修・変更の部分ドキュメントは残っているが，それらを整合させた全体像を把握できるドキュメントがないこと
(3) リリース当時の関係者などが分散して当時の事情を知る人が少ないこと

である。その結果として，プログラム資産がブラックボックス化している。更にモダナイゼーションでは，現行動作を引き継ぐ「現行踏襲」を常に求められる。しかし，そのためには踏襲する機能とは何であって，それがどこでどのように実装されているかを明確化することが必要となる。したがって，モダナイゼーションのプロセスでは，最初に新規開発のプロセスにはない現行システムの把握という工程が必須であり，その結果に基づいて具体的な方式を検討し，決定することが必要となる。[1]

本稿では，上記の難しさに対応して既存システムのモダナイゼーションを支援するために，富士通研究所が提案するシステムの部分特性に合わせた移行アプローチと，これを実現するためのプログラム資産の分析技術を紹介する。

システムの部分特性に合わせた移行アプローチ

業務システム全体を対象としたモダナイゼーションには多大なコストがかかり，また失敗のリスクも高くなることが予想される。そこで富士通研究所は，システム全体から部分を抽出し，それぞれの特性に合わせて望ましい形に移行していくアプローチを提案している（**図-1**）。

例えば，一般的な業務の部分は市販のパッケージやSaaS（Software as a Service）に移行し，変化の少ない部分はそのままの形で継続的に利用する。また，業務が複雑で頻繁に変化する部分は，例えばBRMS（Business Rule Management System）に移行して業務ルールの明確化と変化対応性を上げる。更に，再利用性の高い部分はAPI（Application Programming Interface）を切り出してサービス化し，ほかのシステムやサービスとの連携しやすさを向上させる。

次章以降では，このようなアプローチの実現を支援するために開発した，ソフトウェア地図を生成する技術，業務ロジックの複雑度を定量化する技術，およびプログラムから条件表を抽出する技術，の三つを紹介する。

ソフトウェア地図による全体概観と部分切り出し

ブラックボックス化したプログラム資産を把握する際の最初の課題は，種々の機能と特性を持つ部分が混然一体となっていて分離できていないことである。これに対して富士通研究所は，プログラム資産の全体を概観して，機能ごとの部分を切り出すことができるソフトウェア地図の自動生成技術を開発した。[2], [3]

ソフトウェア地図は，既存システムの多数のプログラム（数万本規模）を入力として，プログラム間の呼び出し関係などのグラフ構造を分析する。そして，密接に関係している部分グラフを一つの機能として抽出することによって，その機能に関連するプログラム群を地図上に自動的に配置して可視化する（**図-2**）。従来，ログ処理などシステム内で共通する処理のプログラムがあると，プログラム間のグラフ構造において複数の機能が共通処理を介して連結されているため，分離できなかった。したがって，機能の自動抽出は困難であった。

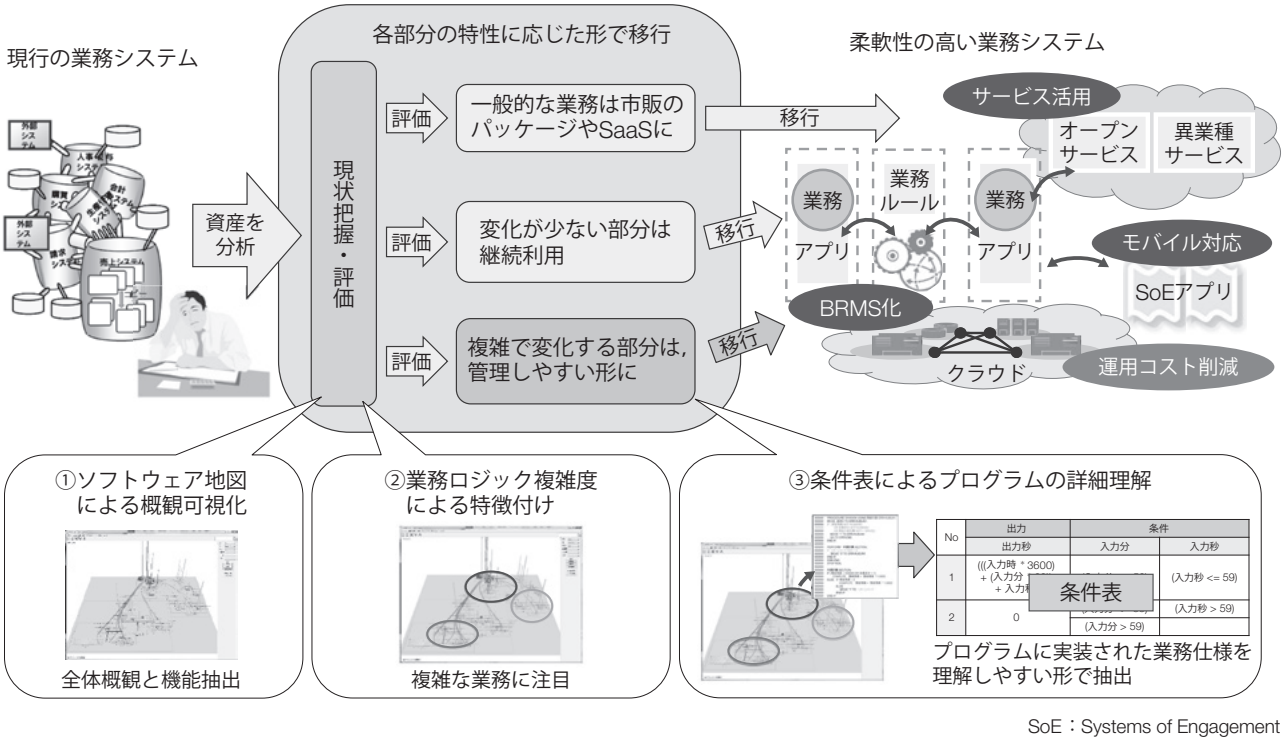

図-1　現行システムの柔軟性を高める移行のアプローチと三つの技術

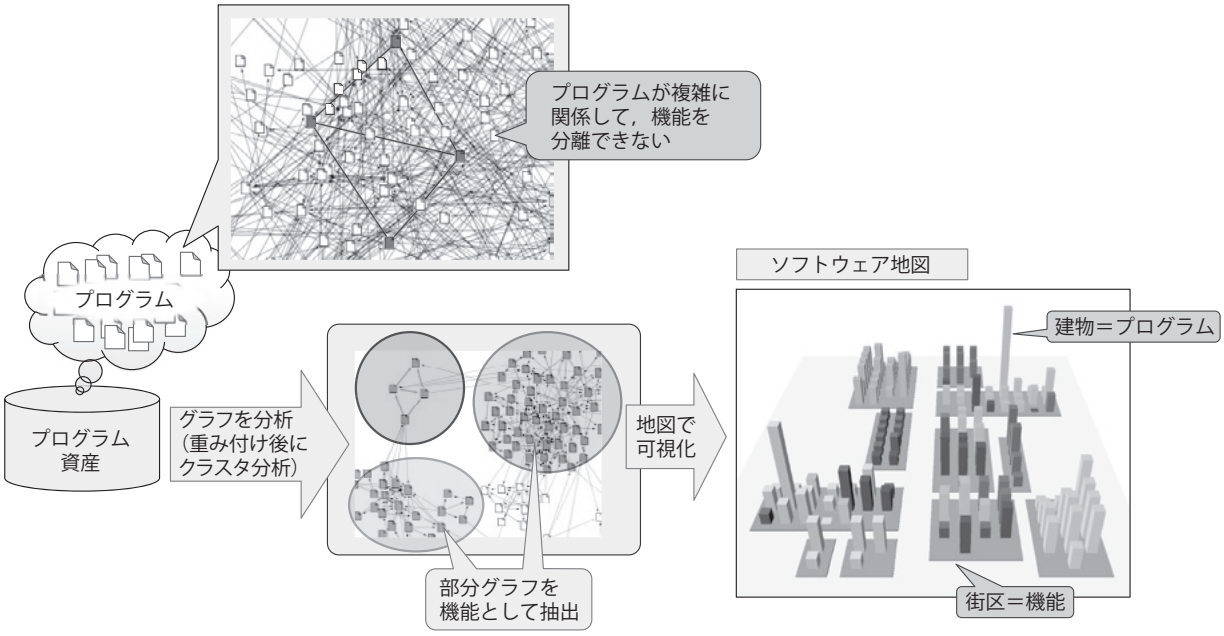

図-2　ソフトウェア地図の生成処理と地図の例

本技術では，共通処理の呼び出し関係の重みを弱めることによって，機能の切り分けを可能とした。ソフトウェア地図は，機能ごとのクラスタとして分類されたプログラム群を市街地図の街区内の建物として表現する。街区が機能に該当するため，街区の個数が抽出された機能の個数となる。更に，地図上で街区間の距離や位置が機能間のそれに対応するように配置する。このソフトウェア地図によって，プログラム資産の全体の概観と機能間の関係を容易に把握できる（図-2）。

業務ロジック複雑度による優先付け

次の課題は，ソフトウェア地図上で抽出したそれぞれの機能の特徴を可視化して，機能部分ごとの移行方針を検討できるようにすることである。これに対して，富士通研究所では，ソフトウェア地図上の建物の高さや色や形に対応するプログラムの特徴を表現し，プログラム資産や機能の特徴を可視化することによって，現行システムの分析者に有効な情報を提供し，現行システムの把握を支援する技術を開発した。

プログラムの特徴は，プログラムの複雑度や行数などのプログラムメトリクス，更新履歴（頻度），品質情報（障害履歴），稼働ログ，およびソースファイルのフォルダ構造などのプログラム資産構造であり，それらを切り替えて表示する。例えば，業務変化の多寡に基づいて継続利用の要否を検討する場合は，プログラムの更新頻度などを建物の高さに割り当て，低い建物を多く含む街区に該当する機能・プログラムを継続利用の候補とする。

プログラムの複雑度とは，プログラムに記述された処理ロジックの複雑さであり，IF文などの命令文の個数や構造から計数できる数値である。従来の複雑度は，プログラム中の全ての命令文を対象とし，業務ロジック（例えば，料金契約の処理）と制御ロジック｛例えば，データベース（DB）を使うための処理｝の両方を含むため，実際の業務自体の複雑度とは一致しないという課題があった。そこで富士通研究所は，プログラム内の業務ロジックだけを識別し，業務ロジック本来の複雑度を定量化する指標として業務ロジック複雑度を提案した。[4] この業務ロジック複雑度を利用することによって，複雑で変化が多い業務に対応する機能部分を選択して，BRMS化を検討できる（**図-3**）。

業務ロジック複雑度では，まずプログラムの外部入力や出力に直接関係する命令文が業務ロジックに関連した実装の部分であるとする。一方，業務色のない文字種判定などのチェック処理やDB制御処理の命令文は，業務ロジックとの関連性が低いとみなして除外する。次に複雑度の値は，上述した業務ロジックに関係する命令文から構成される条件表のサイズとして算出する。後述するように，条件表を正確に算出するには多大なコストがかかる。このため，業務ロジック複雑度では，条件表のサイズの近似値として「条件に関係した項目数」「場合分けの数」「計算式に関係した項目数」

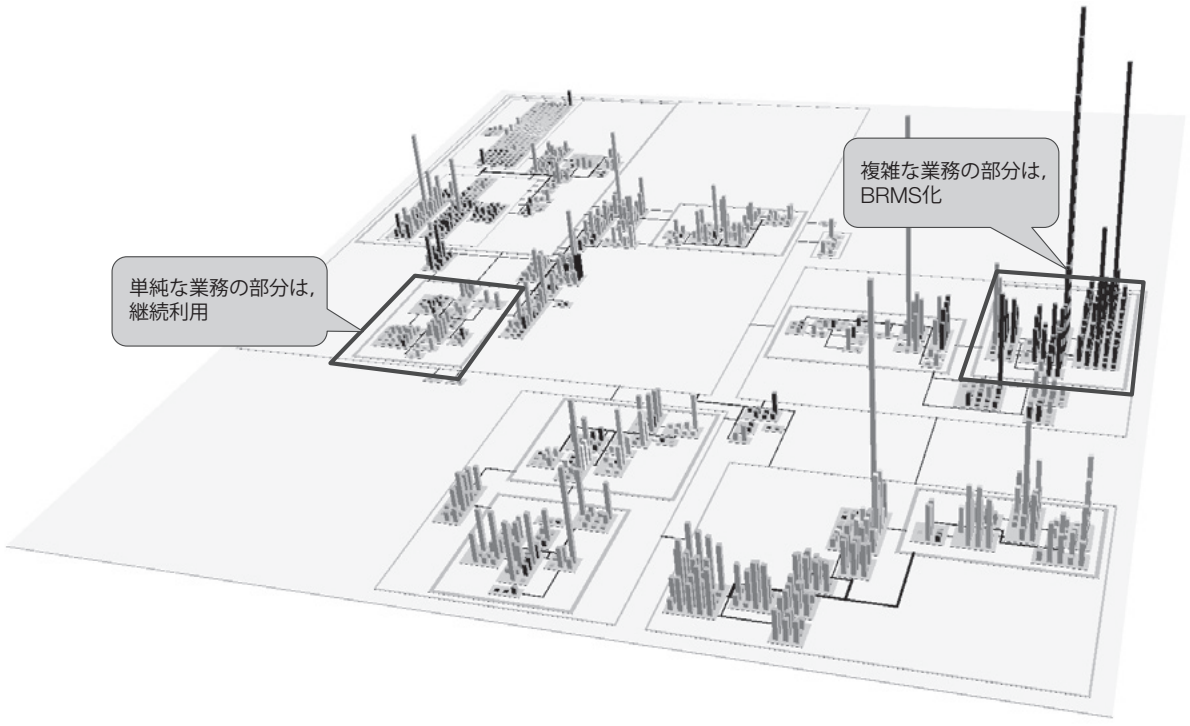

図-3　業務の複雑さを考慮してBRMSなどの移行先を計画した例

の三つの指標を定義して，数万本規模の大規模なプログラム資産に対応した。

図-3に示すように，ソフトウェア地図上の建物の高さに業務ロジック複雑度などの特徴量を割り当てることによって，街区で表現される機能の特徴を可視化できる。また，特徴量の大小比較によって，移行対象として注目すべき機能の優先順位を付けることができる。

条件表の抽出によるプログラム理解の支援

上述した二つの技術により，移行対象として注目すべき機能と，それを構成するプログラム群を特定できた。次の課題は，特定したプログラムの処理内容を詳細に把握することである。これに対して，富士通研究所は，業務システムのプログラムを解析して，そこに実装されている業務上の決まりや計算の方法などを理解しやすい条件表の形式で抽出する技術を開発した。条件表は，プログラムの実行結果であるプログラムの出力が，入力に応じてどのような値になるかを表現する(**図-4**)。

表の出力欄には，出力値が入力や定数の値だけを用いて記述される。また条件欄には，プログラム中のIF文の条件の列挙ではなく，前記の出力値となる場合の条件が論理的に簡略化した形式で記述される。条件表では，プログラムに記述されたサブルーチン呼び出しやGOTOなどの制御構造を解決・除去し，また内部変数を介した値の伝播も厳密に分析する。この特長を活かして，現行システムの分析者がプログラムから自動抽出した条件表を利用することによって，プログラムの処理内容の把握を支援できる。

条件表の抽出は，シンボリック実行技術[5]を用いてプログラムの実行可能な全てのパスを抽出し，それらを整理する方法で行う。ここで，シンボリック実行とは，プログラムの入力変数に10やABCといった具体値ではなく，値が確定しないシンボル値を設定し，プログラムを模擬実行することによって実行可能なパスを抽出する技術である。シンボル値を処理できるように特別に開発した実行環境において，命令文に従ってシンボル値の参照と更新を行い，ループ文のような制御文では記述どおりに実行回数を制御する。

特に，IF文などの条件分岐文において，条件式にシンボル値が含まれるために真偽値を確定できない場合は，条件式の充足可能性を判定し，その

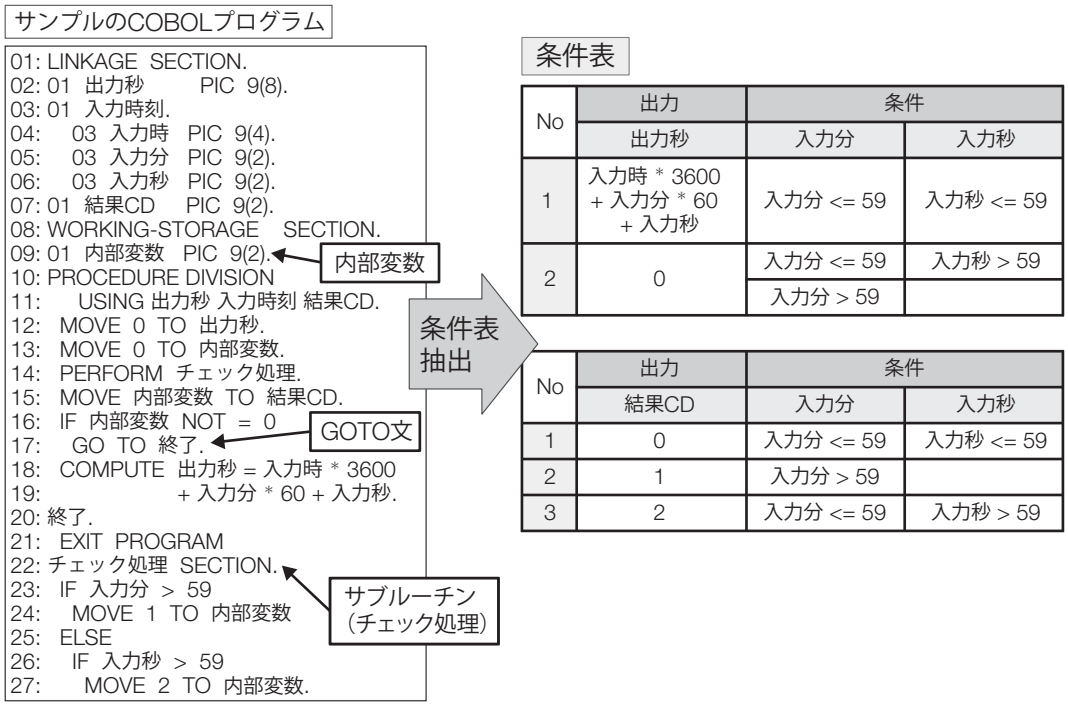

図-4　プログラムの実行結果を記述する条件表の抽出

結果に基づいて実行可能な分岐を選択する。IF文の条件式とその条件式の否定の両方が充足可能である場合は，THENとELSEの両方の分岐が実行可能であると判定する。このような場合，まず一方の分岐として，例えばTHENを選択してパスの追跡を継続する。そして，パスの終点（実行終了の命令文，または最後の命令文）まで到達したら，この分岐点までバックトラックして，もう一方の分岐としてELSEを選択して別のパスを抽出する。このようにして，シンボリック実行によって実行可能なパスを抽出できる。

ここで充足可能性とは，条件式内のシンボル値に適切な値を設定することで，条件式が真となるかということであり，一般的にはSMT（Satisfiable Modulo Theories）ソルバーというツールを用いて判定できる。

シンボリック実行は1970年代から学術研究されてきた技術であるが，実用的なツールの開発や適用は，21世紀以降にコンピュータやSMTソルバーの性能が向上するまで少なかった。富士通研究所は，業務システムの主要な実装言語であるCOBOLを対象としたシンボリック実行ツールSEA4COBOL（Symbolic Executing Analyzer for COBOL）を開発し，[5] テストケースの生成に適用してきた。[6]

SEA4COBOLの別の適用が，COBOLプログラムからの条件表の抽出である。この用途では，プログラムの全てのパスを抽出する必要がある。しかし，プログラムのサイズが大きくなり，条件分岐の個数が増加するほどパスの個数が増加するため，実際の業務システムのプログラムへの適用が難しいという課題があった。これに対して，富士通研究所はまずサイズが大きく分岐が多いプログラムに対して，業務ロジックの複雑さや構造に着目した。処理ブロックの部分に分割して，部分ごとにシンボリック実行技術を用いて条件表を抽出する。次に，サブルーチン呼び出しなどプログラムの流れと，プログラム中の変数の参照や更新を解析した。そして，分割生成した条件表を結合することによって，本来の業務ロジックの条件表を再構成する方式を開発した（**図-5**）。

従来，シンボリック実行は呼び出し元から呼び出し先の順に連続してパスを追跡するため，呼び出し元・呼び出し先の経路数の積程度のパスが抽出され，その個数が非常に大きくなるという問題があった。本方式では，呼び出し元と呼び出し先で別々にパスを抽出するため，それらの個数の和程度のパス数にまで処理を削減できる。この削減効果は，サブルーチン呼び出しが多重であるほど大きくなる。例えば，3重のサブルーチン呼び出

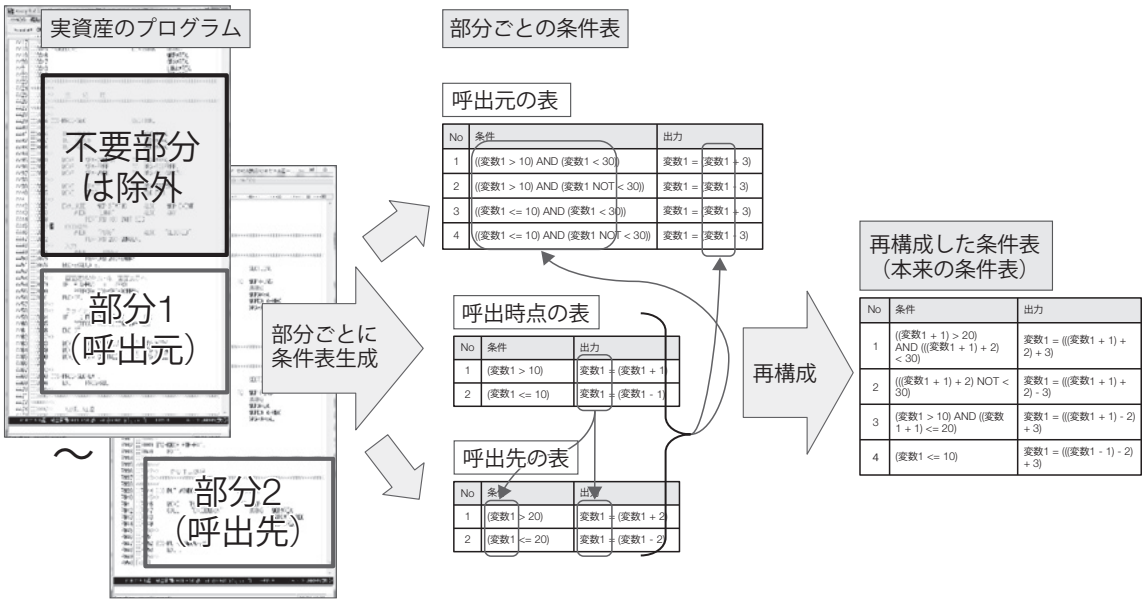

図-5　プログラム部分ごとに抽出した条件表の結合方式

しでは，従来は3,060個の実行経路の分析が必要であったところを41個に削減できた。

抽出した条件表は，プログラムの実行結果を正確に漏れなく表現している。これを使用することで，現行システムの分析者がドキュメントを再整備して移行後のシステムを設計する際に，作業時間の短縮と正確性の向上が期待できる。また，条件表をBRMSルールに加工・変換することによって，業務システムの変化の大きい部分をBRMSに移行する際に，ルールを効率的に作成できる。

む　す　び

本稿では，既存の業務システムのモダナイゼーションにおいて，現行システムを把握するためにシステムを機能部分に分け部分特性に応じて移行するアプローチと，このアプローチを実現するための三つの技術を紹介した。これらのうちソフトウェア地図は，既に富士通のアプリケーションマネジメントサービスの一つとして提供している。

業務ロジック複雑度の定量化は，業務システムのプログラム資産から業務ロジックが複雑なものを選定する社内試行において，SEがプログラムを解読して人手で分析した結果と高い適合率を示すことを確認した。また条件表抽出については，評価実験でプログラムの仕様書を再作成する作業時間を2/3程度に短縮できること，および条件表を加工して実際のBRMSルールに変換できることを確認した。

本稿で示したアプローチと三つの技術によって，SoR（Systems of Record）として安定性や信頼性が求められる一方で硬直してしまった既存の業務システムを分析し，抽出した機能部分の特性に応じて，例えばサービス化，BRMS化，あるいは継続活用のような移行計画の立案が可能となる。更に今後，顧客と企業の間のつながりのシステムとして必要性が増すと予想されるSoE（Systems of Engagement）との連携を図っていくことが可能となる。

参考文献

（1）IPA/SEC：システム再構築を成功に導くユーザガイド.
http://www.ipa.go.jp/sec/reports/20170131.html

（2）K. Kobayashi et al.：Feature-Gathering Dependency-Based Software Clustering Using Dedication and Modularity. ICSM2012, p.462-471（2012）.

（3）K. Kobayashi et al.：SArF Map：Visualizing Software Architecture from Feature and Layer Viewpoints. ICPC2013, p.43-52（2013）.

（4）M. Kamimura et al.：Measuring Business Logic Complexity in Software Systems. APSEC2015, p.370-376（2015）.

（5）前田芳晴ほか：COBOLシンボリック実行によるテストケース生成. ソフトウェア工学の基礎XIX, 近代科学社, p.196-200, 2012.

（6）前田芳晴ほか：業務システム向けの分岐網羅テストケースの生成方式. ソフトウェア工学の基礎XXII, 近代科学社, p.65-70, 2015.

著者紹介

前田芳晴（まえだ　よしはる）
システム技術研究所
基幹系ソフトウェア開発技術プロジェクト
プログラムの分析やテストの技術の研究に従事。

上村　学（かみむら　まなぶ）
システム技術研究所
基幹系ソフトウェア開発技術プロジェクト
ソフトウェアの分析やメトリクスの研究に従事。

矢野啓介（やの　けいすけ）
システム技術研究所
基幹系ソフトウェア開発技術プロジェクト
ソフトウェアの分析や可視化の研究に従事。