特集
Special Feature

［DX（デジタルトランスフォーメーション）：第 2 部　DX の技術と教育，人材育成］

5 DX に向けた既存システム分析・活用の最新技術

基 応 専 般

松尾昭彦 | （株）富士通研究所

DX に向けた関心の高まり

デジタルトランスフォーメーション（DX），すなわちデジタル技術の活用によるビジネスの変革が注目を集めており，企業は競争力の維持・強化のために DX を進め，自らのビジネスモデルを変えようとする動きを強めている．世界のビジネスリーダー 900 人に対する調査[☆1]によると，過去 3 年以内に DX の検討，試行，実践を行ったという回答が 87％を占めている．

ところが，日本では多くの企業が DX に取り組んでいるにもかかわらず実際の変革には結びついていないと言われている．この問題を「2025 年の崖」として指摘した DX レポート[☆2]の公開をきっかけに，老朽化・複雑化した既存の業務システムが DX を疎外する大きな要因であるとされるようになった．

だが，現代の企業はその業務の多くがすでにシステム化されており，その企業独特の業務知識やノウハウが実装された既存システムを捨てて新しく作り直すのは現実的ではない．いかに既存システムを活かしつつビジネスの変革を進められるかが DX 成功の鍵となる．

本稿では既存システムが抱える問題を解説し，既存システムを活かした DX を成功させる上で不可欠な現状分析・連携・移行などを支える最新の技術を紹介する．

☆1 グローバル・デジタルトランスフォーメーション調査レポート 2019, https://www.fujitsu.com/jp/vision/insights/survey3/，富士通（株）．

☆2 DX レポート ～ IT システム「2025 年の崖」克服と DX の本格的な展開～，https://www.meti.go.jp/shingikai/mono_info_service/digital_transformation/20180907_report.html，経済産業省．

既存システムの問題点

企業内で運用されている既存システムの多くは SoR（Systems of Record）と呼ばれ，取引・受発注等の伝票処理を中心とした業務をシステム化しており，企業のビジネスそのものを表す貴重なデータを扱っている．DX が話題となる以前から，このような既存システムのモダナイゼーションが課題とされており，たとえば「システム再構築を成功に導くユーザガイド」[☆3]では，技術面の老朽化，システムの肥大化・複雑化，ブラックボックス化という観点を挙げて調査を行っている．以下に，これらの観点から見た既存システムの問題について述べる．

技術面の老朽化

実装された当時は最新のプラットフォームや言語・ミドルウェアであっても，技術の進歩に伴い新しいものに置き換わっていくため，どうしても技術面での老朽化は避けられない．たとえば，実装時期が古い業務システムでは COBOL が使われていることが多いが，現在では COBOL 技術者が減少する傾向にあって保守を続けるのが難しく，対応できるベンダも限定されるため維持・運用費が高くなりがちになると言われている．また，新しい技術やビジネスモデルに対応することが難しく，全社的なデータ利活用が困難である場合もある．

このような問題は COBOL で実装されたシステムに限った話ではなく，より新しいプログラム言語

☆3 システム再構築を成功に導くユーザガイド　第 2 版，https://www.ipa.go.jp/sec/reports/20180226.html，情報処理推進機構．

© 2020 Information Processing Society of Japan

を使っていても，依存しているアプリケーションフレームワークなどが一般的でなくなると理解できる技術者が減り，保守が困難になってしまうことがある．旧バージョンのサポート切れによるセキュリティ等のリスクも問題となる．

肥大化・複雑化

業務システムは障害対応や業務の変更・追加のため，プログラムの修正がしばしば行われる．このため，システムの規模は次第に増加し，機能も複雑化していく傾向にある．他の機能のデータを参照するなど，設計当初の想定を超えた変更によって修正の影響が思わぬ個所にまで及ぶようになると，確認のための調査やテストを広範囲に実施する必要が出てくる．新機能を開発する際に既存のプログラムを流用して実装することもあり，使われない処理や重複する処理があっても削除や集約を行わずに残してしまうケースも見られる．利用頻度の低い機能があってもシステム部門だけでは必要性が判断できず，いつまでも削除できずに残ってしまうこともある．

このような要因から，既存システムは肥大化していく傾向にあり，修正やテストにますます多くのコストと時間が必要となって迅速なリリースが行えなくなる．

ブラックボックス化

機能追加や修正，障害対応などの案件が続くうちに，ドキュメントの更新にまで手が回らなくなるケースがよく見られる．いったん実態と乖離したドキュメントを更新するには多大な工数が必要になるため，最新の仕様は保守担当者の頭の中にしかないことが多い．このような状況で担当者の異動や退職が発生すると，システム内部がブラックボックス化してしまい，現在の仕様が分からずに影響調査や業務仕様確認，問題個所の洗い出しに時間を要することになる．設計されて久しいシステムでは，要件定義ができる有識者がすでにいなくなり，モダナイゼーションの際に現行仕様の踏襲が難しいという問題もある．

DX 推進のためのアプローチ

DX レポートでの提言を元にとりまとめられた DX 推進ガイドライン[☆4]では，DX の実現やその基盤となる IT システムの構築を行っていく上で経営者が押さえるべき事項を以下の構成で説明している．

DX 推進のための経営の在り方，仕組み

DX 実現のためには経営層自らがあるべき姿（グランドデザイン）を示し，その実現のための体制を整備して責任を持って推進していく必要がある．

グランドデザインを実現するためには，基盤となる体制作りと IT システムの構築が必要となる．DX は全社的な取り組みとなるため，事業部門ごとの個別最適に陥らず全社最適となるように全社的な体制を整備することが必要となる．

DX を実現する上で基盤となる IT システムの構築

IT システムを構築するプロセス（図 -1）では，IT 資産の現状の分析・評価を行い，その分析に基づいて IT 資産の仕分けと移行戦略の立案を行うことが必要になる．また，刷新後の IT システムがビジネスの変化に追従して継続的に進化でき，ビジネスの状況を常に評価できるようにすることが求められる．

既存システム分析・活用のための技術

DX 推進のステップのうち，特に IT システムの構築プロセスにおいては既存システムの分析や活用のための技術が必要となる．以下に，これらの分析

☆4 DX 推進ガイドライン，https://www.meti.go.jp/press/2018/12/20181212004/20181212004.html，経済産業省．

© 2020 Information Processing Society of Japan

や技術について紹介する．

システムの現状の分析・評価のための技術

DX 推進ガイドラインに続き，DX に向けた現状診断のための DX 推進指標が公開されており，約 300 件の企業の自己評価結果も収集・公開されている☆5．しかしながら，この指標は主に経営や推進体制にかかわるもので，この指標だけで既存システムの現状を定量的に把握することは難しい．システムの現状を分析し，定量的な評価や現状理解を支援する技術が必要となってくる．

ブラックボックス化が進んだ既存システムの現状分析は，これまでもモダナイゼーションの際に必要とされており，システムのプログラム資産や保守状況を調査して品質評価を行うサービスが各社から提供されている（図 -2）．システムのどこに保守コストがかかっており，品質上の問題を抱えているのかを把握することは，システムの継続利用の可否を判断する上でも重要となる．

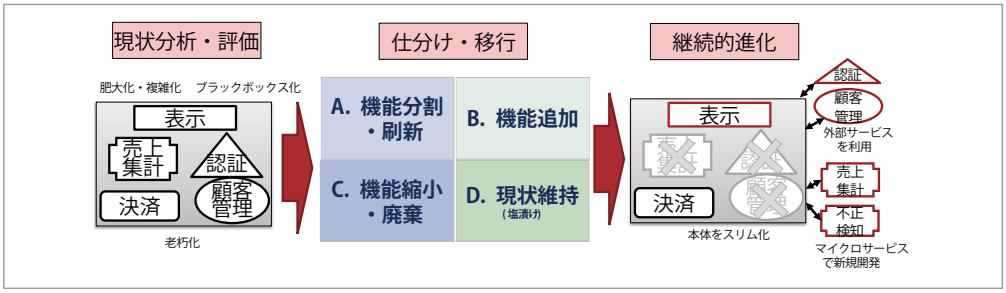
DX に向けた現状分析では，経営視点・業務視点での現状分析と紐づけてシステムを評価する必要があるため，プログラム単位の品質評価だけでなく，業務の観点で全体像を把握し，業務間の関係を分析することが重要となってくる．このため，プログラム間の関係を分析し，その構造を抽出・可視化することで，全体像の把握を支援するような技術が有用である（図 -3）．システム全体の構造を図示し，その上で品質指標などを重ねて表示することで，プログラマでなくても問題個所の把握や移行方針の検討に参加することができる．

さらに，システムの構造だけでなく実装されている機能がどのくらい使われているのか，業務に役立っているのかを定量的に評価することも必要となるため，プログラム資産の分析だけでなく障害対応状況や実行状況を表す動作ログなどの情報，あるいは処理対象のデータなども合わせて分析することも有用である．

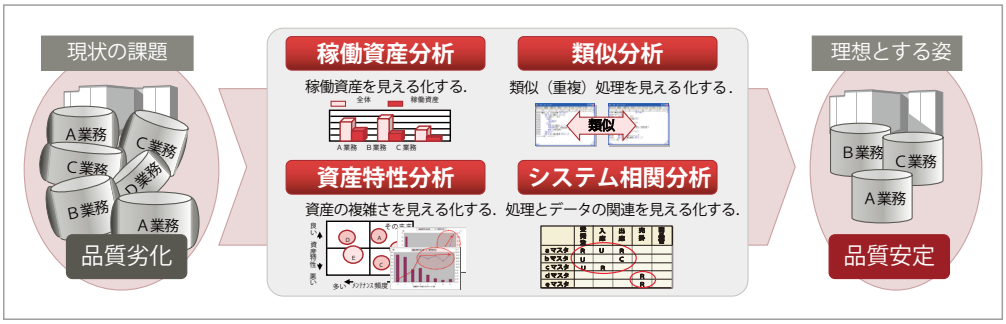
システムの仕分けと移行のための技術

DX レポートでは，現状分析の次のステップとして，機能ごとに評価を行い，頻繁に変更が発生する機能はクラウド上で再構築，新たに必要になる機能はクラウド上に追加，不要な機能は廃棄，変更が不要な機能は塩漬けするなどの方針を決める例が示されている．

これを行うためにはシステム全体の構造や内部の仕様，機能間の関係を把握した上での評価と，現状を踏まえたモダナイゼーションが必要となる．既存システムのモダナイゼーション手法としては，



■図 -1　IT システム構築の実行プロセス



■図 -2　プログラム資産分析サービスの例

© 2020 Information Processing Society of Japan

リホスト・リライト・リビルドが典型的だが，さらにシステムを継続して利用するリインタフェース・ラッピング・リファクタリング・リドキュメント，リプレースなどもあり☆6，適切な手法を選ぶ必要がある．

以下に，それぞれの仕分けのパターンで重要となる技術や手法を紹介する．

機能分割・刷新

既存システムは修正に時間とコストがかかることが多く，頻繁に変更が発生する機能は切り離して新しいプラットフォームに移行し，変更しやすくすることが望ましい．

しかし，既存システムの多くは肥大化・複雑化が進み，多くの機能が複雑に絡み合い一体となった構造となっているので，1つの機能だけを切り離すのは容易ではない．機能間の関係を詳細に解析し，リファクタリングによって関係を整理した上で，必要な関係をインタフェースとして実装する必要がある．

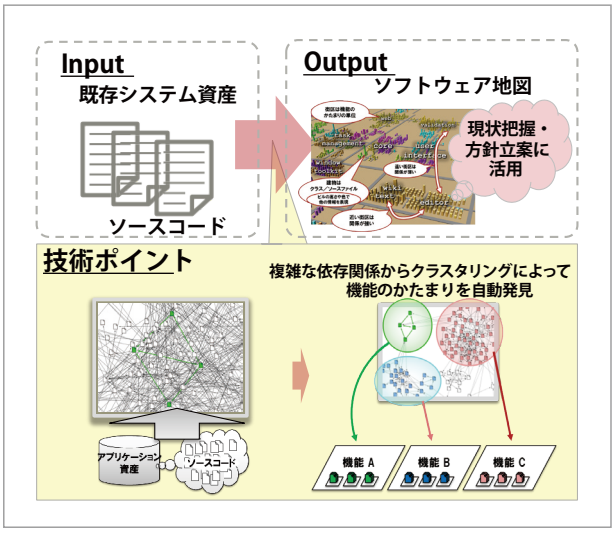
このような分割を実施するためには，システムを構成するプログラム資産を分析し，プログラム間やデータ・画面間などの相関関係を抽出する技術が有用であり，現状分析のサービスにおいて提供されていることも多い．また，より高度な分析として，分割しやすい境界，すなわちインタフェースが少なくなる境界を自動的に見つけ出す疎結合化分析という技術もあり，業務観点だけでは分割が難しい既存システムに対して有用である（図 -4）．

機能を新しいプラットフォーム上に移行する際には，現行の仕様を再現する必要があるが，ブラックボックス化したシステムでは最新の仕様がドキュメント化されていないことが多い．このため，現行システムのプログラムを機械的に変換するか，実装されている仕様を抽出して再現する必要がある．

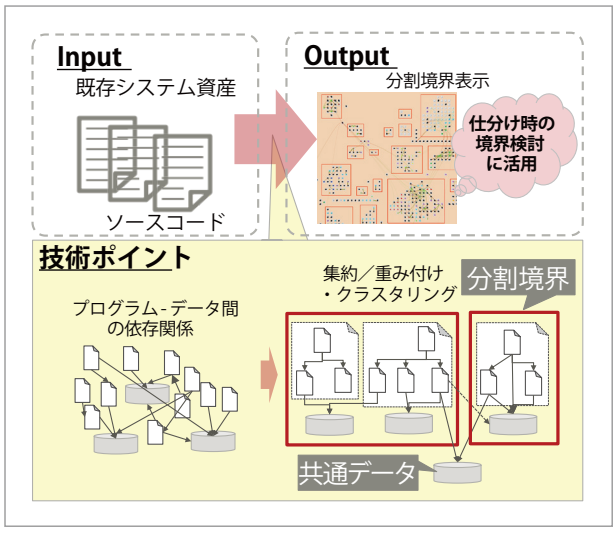
プログラムを変換して移行する手法はリライトと呼ばれ，その実現方法は実装言語や動作環境に大きく依存する．既存システムで多く見られる COBOL から Java などの新しい言語への変換に対するニーズは高いが，言語仕様がまったく異なるため，機械的に変換すると不自然で可読性が低いプログラムになってしまうという問題がある．このため，このような変換サービスを提供するベンダは変換仕様を個別に調整することで対応していることが多い．

機械的な変換が難しい場合，リドキュメント，すなわち実装されている仕様をプログラムから復元してドキュメント化し，この仕様に基づいて新しいプログラムを実装することが考えられる．

☆6 モダナイゼーションの実践事例，https://xtech.nikkei.com/it/atcl/column/15/040200056/040200002/，（株）日経 BP.



■図 -3 ソフトウェア可視化技術の例



■図 -4 疎結合化分析技術の例

© 2020 Information Processing Society of Japan

機能追加

既存システムにデジタル技術を活用した新しい機能を追加する場合，老朽化したプラットフォームでそうした機能を利用するのは難しいため，既存システムの外に新しい機能を実装し，両者を連携させるのが有用である．このためには，既存システムの機能を API（Application Programming Interface）として呼び出せるようにするか，既存システムのデータを新機能側からアクセスさせるための仕掛けが必要となる．

既存システムの機能を API として呼び出すためにはインタフェースを実装する必要があるが，システムの変更が難しい場合，すでに持っているインタフェースを変換して呼び出せるようにするラッピングという手法が有効であり，メインフレーム上のシステムを WebAPI として呼び出せるようにする製品や，既存システムの画面インタフェースを API 化する製品などが提供されている．

逆に，既存システムから新機能を呼び出す必要がある場合，CDC（Change Data Capture）と呼ばれる技術を用いればシステムを変更しなくてもデータベースの更新をトリガにして新機能を呼び出すことができるので，比較的容易に連携を実現できる（図 -5）．ただしデータの更新処理を行う場合は既存システム側の対応も必要となる．

機能縮小・廃棄

既存システムは肥大化していることが多く，利用されていないプログラムも含めて使い続けたり移行したりするのはコストと時間の負担となる．

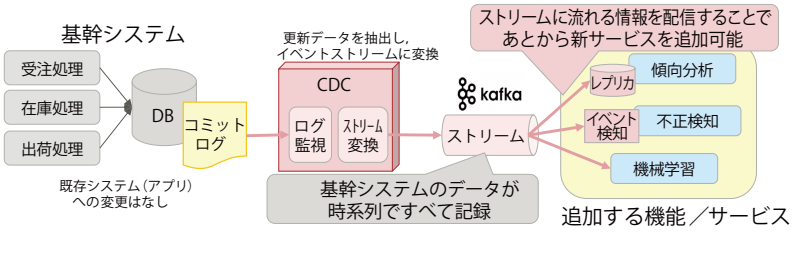
このため，稼働資産分析と呼ばれる手法によって使われなくなったプログラムを特定し，取り除くことが有用であり，これによって肥大化が進んだシステムでは 30 ～ 40％のスリム化が行えることもある．また，動作実績のないプログラムを稼働ログ等から特定したり，業務を見直して利用頻度が低い画面や帳票を削減することで，より効果的なスリム化が実現できるが，この判断のためには業務部門も巻き込んだ分析と判断が必要となる．

さらに，必要であっても他社との差異化要素がない業務はパッケージソフトウェアや SaaS（Software as a Service）でリプレースして，自社の強みとなる競争領域に集中することが望ましい．このためには，現行機能とパッケージなどが提供する機能との違いを把握する Fit&Gap 分析を行った上で，違いにどう対応するか業務的な判断が必要となる．

現状維持

既存システムをすべて置き換えるのではなく，更新が発生しないような機能は現行のまま運用を続けることが合理的である．運用コストが問題となる場合，アプリケーションは変更せずに実行環境を仮想化してプライベートクラウドやパブリッククラウドに移行させるリホストと呼ばれる手法があり，移行に必要な情報を自動的に収集して仮想化を支援する機能を提供しているクラウドベンダもある．ただし，既存システムが動作している OS やミドルウェアがサポートされていない場合はこの手法は適用できない．また，パブリッククラウドへの移行ではデータ漏洩などのセキュリティリスクに注意する必要がある．

DX においてはデータの活用が重要となるので，システムとしては現状維持のままでも，個々のシステムのデータを統合して活用する基盤を導入することが望ましい．データの参照のみであればリプリケーションで実現できるが，更新が必要な場合は，既存システム側



■図 -5　CDC（Change Data Capture）技術の応用例

© 2020 Information Processing Society of Japan

の更新処理を呼び出すなどの仕組みが必要となる．また，データの標準化などの処理が必要になる場合もある．

継続的進化の維持のための技術

刷新後のシステムが DX で求められる変化のスピードに対応できるように，変化に対応しやすいマイクロサービスアーキテクチャを採用するのが効果的と言われている．開発体制も，従来型の大規模開発ではなく小規模なチームによるアジャイル開発にし，リリース手順を自動化する CI/CD（Continuous Integration/Continuous Delivery）環境によって高速なリリースを実現することが望ましい．OS などの修正パッチへの対応の自動化や潜在バグの自動検出・修正などを行う技術も実用化が進んでおり，これらを適用することで開発者の負担軽減が期待できる．

DX の実現に向けて

DX の実現に向けては，本稿で取り上げたような個々の既存システムの移行だけでなく企業全体のアーキテクチャをデザインしていく必要があり，このためには多数の業務システムが絡み合った企業システムの全体像を把握することも必要になる．また，移行後のシステムではビジネスの状況を常に把握できることが求められるため，データや動作状況を全社的にモニタリングするような仕組みも必要となる．これらの実現には全社的な取り組みを継続していく必要がある．

2025 年の崖を超えるためには，既存システムの現状を踏まえた上であるべき姿に向けて実現可能な移行パスを定め，実行していかなければならない．本稿で紹介した技術が DX 実現の助けになれば幸いである．

（2020 年 7 月 31 日受付）

■松尾昭彦（正会員） a_matsuo@fujitsu.com
（株）富士通研究所サービスビジネス開発・運用ユニット．業務システムの保守や再構築に関する研究開発に従事．

© 2020 Information Processing Society of Japan