URL https://journal.ntt.co.jp/article/31212
DOI https://doi.org/10.60249/25011201

NTTデータグループ
技術革新統括本部　A&D技術部

竹之内啓太　Keita Takenouchi

「プログラム合成」でソフトウェアマイグレーションの効率化を図る

デジタルテレビの映像データ複号や自動車のクルーズコントロール等の制御をはじめ，世の中のあらゆるモノにソフトウェアが組み込まれています．ソフトウェアの開発は，かつては要件定義，設計，プログラミング，試験等の工程を人手で行う労働集約型の環境で行われてきました．その後，ソフトウェアの部品化による再利用，プログラミングや試験の自動化等が行われるようになり，最近では生成AI（人工知能）も活用されるようになりました．新規にソフトウェア開発をする場合は，こうした新しい取り組みによる効果が大きい一方で，システム更改等では既存のソフトウェアの資産流用，改変等の制約を伴うことが多く，新規開発と同程度の効果は必ずしも期待できません．入力データと出力データを定義することで自動プログラミングを行う「プログラム合成」により，システム更改等におけるソフトウェアマイグレーションの効率化に挑む，NTTデータグループ 技術革新統括本部 竹之内啓太氏に，ソフトウェアマイグレーションの効率化，そして研究者と開発者の間の意識ギャップを埋める思いを伺いました．

入力データと出力データからプログラムを生成する「プログラム合成」

現在，手掛けている業務の概要をお聞かせいただけますか．

「プログラム合成」を用いたマイグレーション技術をテーマとした開発に取り組んでいます．

プログラム合成技術は，自動プログラミングの一種です．通常のプログラミングでは，設計書等に従って人間がソースコードを作成し（プログラミング），机上およびシステム等に搭載してテストを行うことで運用に供します．最近では，設計情報を入力すれば生成AI（人工知能）がプログラムを生成する自動プログラミングも登場しています．一方，プログラム合成は入出力データを先に用意（定義）することで，それを満たすプログラムを自動生成する技術です．データベースシステムを例に説明すると分かりやすいかもしれません．一般的なプログラミングでは，入力データから出力データを得るためのロジックを人間が考えて設計・ソースコードを作成し，テストにおいてはテストデータを入力して，出力が期待どおりであるかをチェックします．プログラム合成では，入力データと出力データを先に用意して，この入力データから出力データをつくるようなプログラムを自動的に生成するもので，あらかじめデータを用意するだけでプログラムを生成できます．

プログラム合成技術の開発において，データベースからの情報抽出等，データベースを操作する言語であるSQL（Structured Query Language）の自動生成をテーマに取り組んできました．入力と出力を対応付けるSQLクエリの構文要素の組合せを探索するアルゴリズムを考案し，入力データと出力データの対応条件に合致するものを選び出すことで，意図するプログラムが生成されます．私たちが開発したツールでは，ほとんどの場合10秒以内，場合によっては1秒かからないほどの時間で複雑なSQLクエリを自動生成することができ，こういった技術はこれまでにはないものです．この結果は，入出力例からSQLクエリを自動生成するアルゴリズムの開発において，最先端の手法を関係代数の性質に基づいて拡張した新規性により，提案手法が既存手法の性能を大きく上回ることを実験的に示しました．このことが評価され，データベース分野の世界トップカンファレンスである，International Conference on Very Large Databases（VLDB）2021において論文採択されました（平均採択率：18.6%，全212論文のうち日

本からの採択は 3 件のみ）.

なお，「生成するプログラムの構造が複雑すぎる場合，生成にかかる時間が膨大になる可能性がある」「入出力値の対応関係が推測しづらいものほど自動生成としてサポートするのが難しくなるという傾向がある」という特徴があり，SQL構文要素には対応済みのものと未対応のものがあります（**図 1**）.

新しい概念のプログラム合成をどのような分野に応用しているのでしょうか.

既存の大型のデータベースシステムの更改時や機能拡張・追加時におけるソフトウェアマイグレーション等に活用しています（**図 2**）.

一般に，機能を保ったままプログラムを別のプログラムに変換する技術には，「ルールベースの変換」「言語モデルベースの変換」の 2 つのアプローチがあります.

「ルールベースの変換」は，人間が変換ルールを定義しルールを適用するもので，（理論上は）変換の正しさが保証される反面，ルールを定義するのが困難，変換ルールでサポートされない構造があると変換に失敗する，という特徴があります.

「言語モデルベースの変換」は生成AIによるもので，AIが大量のプログラム集合から変換ルールを学習し変換します．人手の手間が小さい一方で，学習データが必要，変換の正しさは保証されない，現在のレベルでは扱うデータ項目数の多いシステムには不

☑ 対応済み SQL 構文要素

- SELECT
- WHERE（条件式：等号，不等号，AND，OR）
- GROUP BY（MAX，MIN，SUM，COUNT 等）
- JOIN/LEFT JOIN
- DISTINCT
- ORDER BY
- HAVING
- ウィンドウ関数(RANK 関数，MAX 関数等)

※ 生成するプログラムの構造が複雑すぎる場合，生成にかかる時間が膨大になる可能性がある

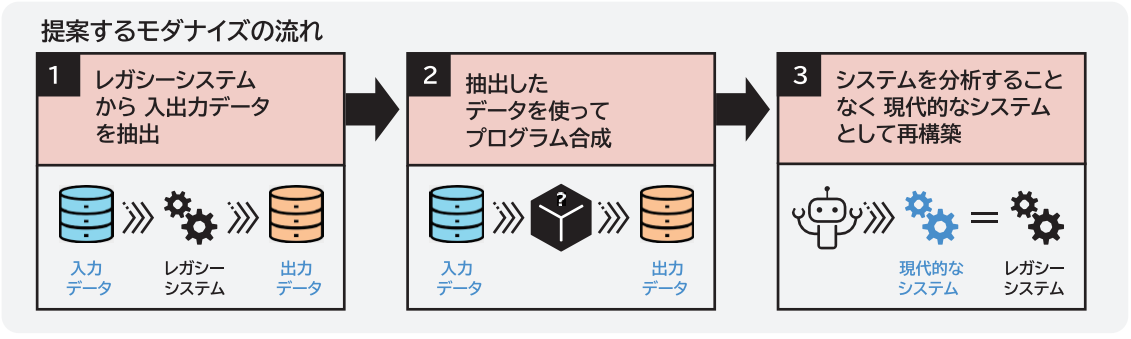
☒ 未対応の SQL 構文要素

- UNION
- 一部の条件式（LIKE，EXISTS，文字列どうしの比較）
- 四則演算
- 日付計算
- 文字列操作（SUBSTR など）
- CASE 式
- ユーザ定義関数
- …

※ 傾向として，入出力値の対応関係が推測しづらいものほど自動生成としてサポートするのが難しくなる

図 1　自動生成できる SQL クエリの構文要素

入出力データに基づいてプログラムを再構築することで，
ブラックボックス化した既存資産の分析を必要としないマイグレーションを実現する



特に COBOL から SQL への変換など，プログラムの構造が大きく異なる変化する場合に
他の自動化手法(ストレートコンバージョンなど)を上回る効果が期待できる

図 2　プログラム合成技術を活用したマイグレーション

向き，といった特徴があります．

私の提案した「合成ベースの変換」は，入出力データが保たれるようにプログラムを自動で復元するもので，人手の手間が小さい，変換の可否がプログラムの構造に非依存といったメリットがある反面，合成アルゴリズムが必要，変換の正しさは保証されない（入出力データの範囲では正しい）といったデメリットがあります．現在取り組んでいる「合成ベースの変換」によるマイグレーションは，例えば，COBOL（プログラミング言語）で書かれた既存の古いシステムから入力データと出力データを抽出します．その入力と出力のデータを使ってプログラム合成をかける際に，COBOLの動作をSQLでつくり直すといったように，新しいシステムとして現代的なテクノロジを使ってつくり直すこと（モダナイゼーション）が１つのねらいとなっています．もちろん動作が完全に一致するものが生成される保証はありませんが，もともとブラックボックス化していて手のつけられなかったようなシステムを部分的にでも新しいシステムとしてつくり直す，というようなところで実際に技術活用しています．

この考え方によるプログラム合成のマイグレーションへの適用スコープ例を**図３**に示します．

データベースを持つ現行のシステムにおいて，データは上流システムから供給され下流システムへ提供されます．マイグレーション後のシステムはデータウェアハウスを持ち，マイグレーションプロジェクトにおいて，現行のデータベースのデータをデータウェアハウスに移行します．マイグレーション後，下流システムはこのデータウェアハウスからデータを読み込みますが，データベースとデータウェアハウスのデータ形式には差があるため，適切にデータを読み込むにはこのデータ形式の差を吸収する必要があります．この差の吸収は，

1．上流システムからデータベースとデータウェアハウスの両方に同時にデータを供給するように構成し，これを一定期間稼動させる．

2．データベースとデータウェアハウスに格納されているデータを入出力データとしてプログラム合成を実行することで，データベースとデータウェアハウスのデータ形式の差を補完するためのSQLを生成し，マイグレーションを行う．

といった手順で対応します．データウェアハウスから連携用データを抽出する部分は，入力データと出力データが確定しており，プログラム合成が適用される部分なのです．

さて，NTTデータのシステム開発案件にはマイグレーションを伴うものが多くあるため，当面はプログラム合成技術をブラッシュアップしながら各案件のマイグレーションへの適用を図っていくつもりです．それ以外のユースケースとして，データ分析におけるSQLクエリ生成が考えられます．現在は，データ分析に際して人手により対応するSQLクエリを作成しているのですが，SQLクエリを作成できる人はそれほど多くはいません．とりあえず欲しいデータを手で作成し，プログラム合成でSQLクエリを生成する，さらにその過程をいくつか試して，どのようなSQLが生成されるか等を確認することで，SQLの実践型研修への応用も可能になると思います．

また，NTTデータはグローバルカンパニーをめざしており，この観点から日本でプログラム合成の成熟度を高めることで，グローバル展開を図ることができるのではないかと思います．そして，海外でうまくいった事例を日本にも取り込む等，ノウハウの共有が可能になると思います．

プレプリントや発表論文一覧でトレンドウォッチを欠かさず行う

開発者としてのスキルはどのように磨いているのでしょうか．そして，どのように業務に活かしているのでしょうか．

私は，2017年にNTTデータに入社してソフトウェア工学の専

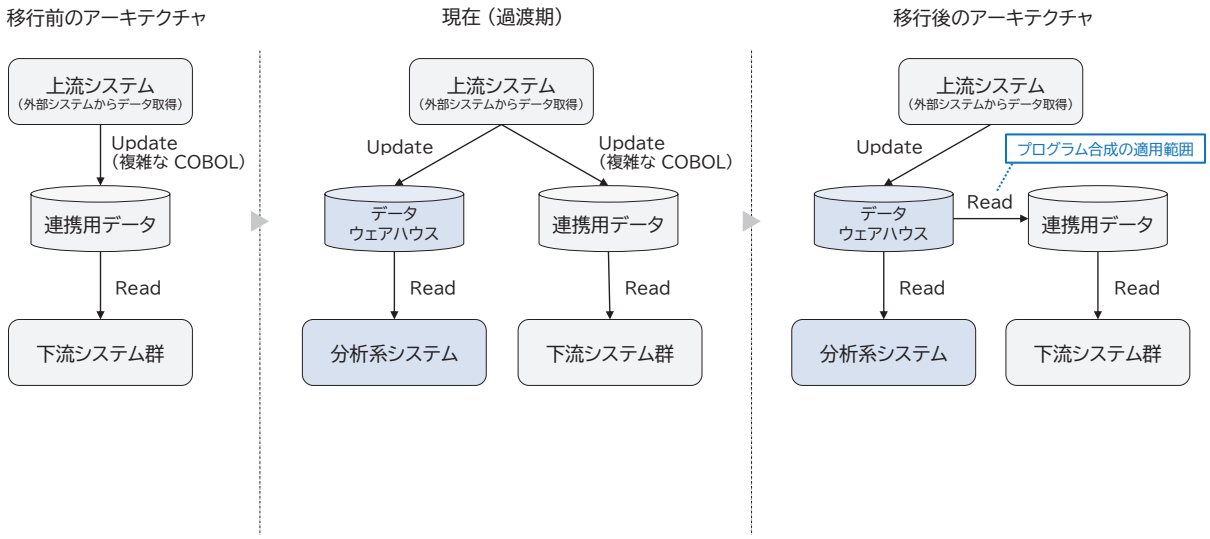


図３　プログラム合成のマイグレーションへの適用スコープ

門家として，国際会議や情報処理学会等の学会へ論文を投稿するような研究から，技術を現場に適用するフェーズまでの一連のプロセスに携わってきました．最初は，レガシーシステムのモダナイゼーションの支援部隊で，何種類かあるモダナイゼーションの案件の特性を分析して，最適な技術の適用支援を行ってきました．そして，この方法をドラスティックに変えることをめざして，プログラム合成を提案し，それをテーマに研究を始めました．

研究のフェーズでは，コミュニティとのコンタクトを常に取りながら，学会への参加，論文投稿等を続けています．そのために，学会や国際会議の新規発表論文は毎日読んでいます．さらに，論文は査読から発表までに半年以上の時間を要するので，論文査読前に発表されるプレプリントや発表論文の一覧を注意深くチェックして，どういう方向に研究が進んでいきそうなのかといったトレンドウォッチは欠かさず行い，スキルを最新の状態にキープしています．ここ3年ほどで一気に注目を集めている生成AIについては進化も激しいため，このトレンドウォッチがより重要になるでしょう．

一方で，新技術の実システムへの適用においては，トレンドウォッチしながら，短期的なメリットを見出すのか，中長期的なメリットを見出すのかを分類したうえで，現場に足を運んで課題を調査分析し，適用技術を検討します．プログラム合成を活用したモダナイゼーションの対象となるシステムは，比較的規模の大きなシステムが多くあります．大規模なシステムでは，枯れた技術を使う傾向にあるため，新技術の導入に際してはその意識ギャップを埋めることも必要になります．そこで，プログラム合成の適用の場合については，部分的に使ってみて従来と比較してかなり効率的になるといった価値を理解してもらい，徐々にスケールさせています．

研究者とソフトウェア開発者の意識のギャップを埋める

事業会社は異動がありますが，こうしたスキルを将来的にどう活かしたいとお考えですか．

NTTデータに入社以降，研究者としてコミュニティへの参加や論文作成等を行い，それと並行して，成果をソフトウェア開発に応用していくというプロセスを経験してきました．こうした経験の中で，大学や研究所のソフトウェア工学の研究者と，企業のソフトウェア開発者との間の意識のギャップを感じています．

ソフトウェア工学の研究者は，コミュニティ活動や論文発表等の観点から，斬新で洗練された手法や正しさが証明できる技術をつくることに意識が向いています．一方でソフトウェアの開発者は，工程管理，進捗管理，予算管理等のプロジェクトマネジメントの観点から，目の前の課題に意識が向いています．こうした意識の差があるうえに，それぞれの活動の場が異なっているため，両者の接点がほとんどなく，それがギャップにつながっていると思います．

この開発者と研究者のギャップを埋めるために，例えば現場の課題を解くにあたって最新の知見を駆使して取り組み，新しい手法を適用して得られた知見を研究コミュニティ側にフィードバックするといった橋渡しが必要だと思っています．これまで私は，これに近い立ち位置で業務に携わってきたので，その経験を活かしてとりあえずは研究のコミュニティ側に軸足を置きつつ，実際の開発現場との間を橋渡しする役割を担ってみたいと思います．

後進や読者へのメッセージをお願いします．

事業会社にいるとビジネス関連の部分を直視することが重要になるため，研究や技術，そしてそのトレンドに関して意識が向く機会が少なく，ノウハウや知見の蓄積や継承も手薄になりがちです．逆に研究者はコミュニティとかかわる機会も多く，論文投稿や学会発表をとおして，ノウハウや知見の蓄積は日常的に行われています．しかし，コミュニティの中に閉じてしまうことで，実際の現場で何が起こっているかといった部分へ意識が向かなくなり，現場との距離が遠くなります．そこでもし，事業会社の現場と研究者の接点をとることができれば，それぞれが手薄になっている部分を補完し合うことができるのではないでしょうか．最近は，ビジネスも研究も他者と連携することが多くなっているので，お互いをこの連携の相手と意識していくことで，うまく接点ができると思います．

お客さまやパートナーの皆様にとっても，生成AIの登場によりソフトウェアの開発プロセスが大きく変化しようとしています．こうした流れの中で，私たちはソフトウェア開発プロセスや技術に関して積極的に発信していくことを検討していますし，実際に多くの開発案件で新しいソフトウェア開発方法を適用しているところです．新しい技術をキャッチアップし，それを現場で使っていくことを一緒に行っていただきたいと思います．