NTT DATA Technology Foresight 2012

高速デリバリを実現するソフトウェア開発自動化

ソフトウェア開発が大規模化・複雑化する一方で、社会環境やビジネス環境の著しい変化に対応するため開発期間は短期化している。人海戦術での対応には限界が見えつつあるなか、生産性を飛躍的に向上させることで、劇的な短納期を実現する「ソフトウェア開発自動化」の技術に再び期待が寄せられている。本稿では、そのようなソフトウェア開発自動化の技術動向と当社の取り組みを紹介する。

ソフトウェア開発自動化の必要性

近年のグローバル化等によるビジネス環境の急激な変化により、それを支える IT システムの「開発期間の短期化」への要求が年々高まっている。その一方で、IT システムに求められる機能は高度化しており、そのため開発しなければならないソフトウェアは大規模化している。大規模化する IT システムを求められる期間で開発するため、これまでは人海戦術での対応で凌いできたが、そのような人に依存した方法だけでは、品質のばらつき、管理コストの増大を招くなど、すでに限界を迎えつつある。よって、これまで以上に開発者 1 人が生産できるソフトウェア量を拡大することで開発短期化の要求に応える必要がある。ソフトウェア開発自動化はそのための有力な技術となる。

ではなぜ今ソフトウェア開発の自動化なのか。そこには IT 要素技術(CPU やストレージ等)の劇的な進歩がある。ソフトウェア開発の自動化は以前からその取り組みはあったが、CPU やメモリ等の IT リソースの制約を考慮する必要があり、それが自動化を困難にさせていた。現在は、IT 要素技術の進歩により、IT リソースが潤沢に低コストで利用できる環境ができつつあり、それがソフトウェア開発の自動化を再考するきっかけとなっている。

8 つのソフトウェア開発自動化技術

ソフトウェア開発自動化は古くから CASE(Computer Aided Software Engineering)ツールや MDA(Model Driven Architecture)ツールなどの形で提案されてきたが、特にプログラムを自動生成する技術の実現にチャレンジしてきた歴史がある。しかし、ソフトウェア開発の自動化はプログラム自動生成だけではない。ソフトウェア開発が単なるプログラム作成ではなくなった今、開発を構成するさまざまな作業の自動化を考えることが必要になっている。ソフトウェア開発の自動化には大きく 8 つの領域があると考えられる(図 1)。

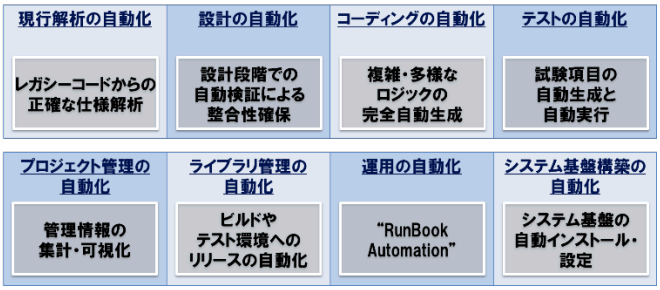

図 1：ソフトウェア開発自動化の 8 領域

現行解析の自動化は、リバースエンジニアリング、ソフトウェアを再構成するために行う調査作業の自動化である。レガシーシステム更改の必要性や保守の効率化要請から近年ニーズが非常に高まっている。

設計の自動化は、設計作業および設計レビューの自動化である。以前から設計支援のための設計エディタ(UML エディタ等)は多く存在するが、自動化に関する本格的な取り組みはあまりない。

コーディングの自動化は、プログラム作成作業の自動化、プログラム自動生成である。近年は、多種のツールが存在するため目的に合わせて適

用することが重要となっている(後述)。

ライブラリ管理の自動化は、プログラムのビルド作業やテスト環境へのディプロイ作業などの自動化である。Continuous Integration(継続的なインテグレーション)、Continuous Delivery(継続的なデリバリ)といった概念で現在非常に注目されている。

プロジェクト管理の自動化は、システム開発におけるプロジェクト管理情報の集計・可視化の自動化である。最近の動きとして、IBM 社主導で管理ツール間の自動連携をはかる OSLC(Open Services for Lifecycle Collaboration)標準化が進められている。

運用の自動化は、システム開発後の運用段階における IT システムの運用作業の自動化である。IT 運用管理に関するプロセスを自動化する「RunBook Automation」といった概念で注目され始めており、運用管理ツールベンダ各社が自動化機能を強化した製品開発に力を入れている。

システム基盤構築の自動化は、各種ハードウェアやネットワークの設定、ミドルウェアのインストール・設定などの自動化である。仮想化、クラウドコンピューティングの発展にともない、システム基盤はプログラマブルになってきており、それが自動化を推し進めている。

以降では、「コーディングの自動化」、「テストの自動化」の領域の自動化について、その動向と NTT データの取り組みを紹介する。

コーディングの自動化技術

コーディングの自動化、つまりプログラムの自動生成技術は、それ自体は目新しいものではなく、これまでも様々なツールが開発されてきた。初期の取り組みは失敗に終わったものも多いが、現在は計算機能力の向上と適用技術の成熟化により十分な実用性を持ったツールが多く存在する。

現在の自動生成ツールは、自動生成ツールの開発対象となるアプリケーションの性質により、大きく 3 つの種類に分類できる(図 2)。開発対象のアプリケーションに期待される寿命と要件の自由度で分類される。アプリケーションの寿命は、使い捨てに近いものから長期間の保守を要するものまで考えられるが、一般的に、長期間の保守が必要なアプリケーションに対しては、設計とプログラムの同期やコードと要件のトレーサビリティ確保等のための仕組みが求められる。要件の自由度は、自動生成できるアプリケーションの多様性と言い換えることができる。自由度が大きいほど多様なアプリケーションの自動生成が可能であるが、一般的に、自由度が小さいほど固定範囲(再利用範囲)が大きくできるため、ツールの生産性効果は高くなる傾向がある。

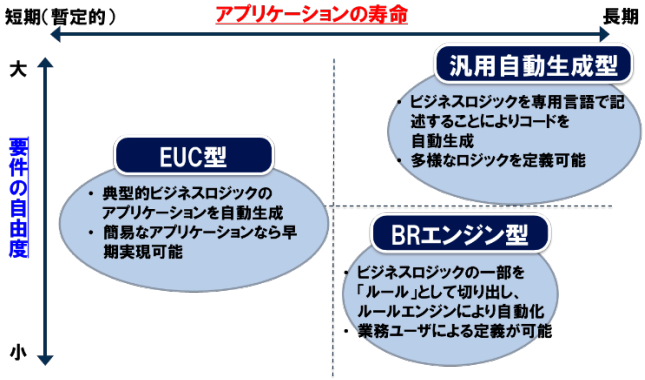

図 2：プログラム自動生成ツールの分類

EUC(End User Computing)型は比較的単純なアプリケーションを超高速に作成するための自動生成ツールである。試行錯誤を前提としたアプリケーション(データ分析等)や短期間利用のアプリケーション(キャンペーンサイト等)に適する。

BR(Business Rule)エンジン型は特に変更頻度が高い部分のみを自動生成することで、変更開発を高速に行うための自動生成ツールである。変更頻度が高い部分はユーザが自ら編集できる抽象度の高い表現のビジネスルールとして切り出す。一部のビジネスルールが複雑かつ変更が頻繁

に発生するアプリケーション(特に金融・保険の商品管理等)に適する。

汎用自動生成型は業務アプリケーションに特化した専用言語により設計情報を記述し、その情報からプログラムを自動生成するツールである。業務アプリケーションの 100%自動生成が可能で、設計情報とプログラムの同期を維持できる仕組みを持つ。長期保守を要するアプリケーション開発に適する。

先述のとおりツール自体は成熟化しているため、これら様々なタイプのツールを適材適所にどのように活用するかが成功のポイントとなる。

汎用自動生成型のツールの例として NTT データ TERASOLUNA ViSC を紹介する。TERASOLUNA ViSC は開発対象のソフトウェアに合わせてカスタマイズして利用することを前提としたツールである。カスタマイズすることにより対象ソフトウェアのプログラムを 100%自動生成できるようにする(図 3)。開発者はカスタマイズされた専用ツールを利用してソフトウェアの設計情報を投入することにより、プログラムやレビュー用の設計書を完全に自動生成することができる。完全自動生成の利点は、設計情報とプログラムが完全に同期できることである。設計情報とプログラムが乖離すると、保守フェーズにおいて仕様理解や修正範囲の特定が難しくなるといった問題を引き起こす。設計情報とプログラムの完全同期によってそのような問題も解消される。一方で、導入時にツールをカスタマイズすることが必要となるため、適用までに一定の準備期間・コストが必要となる。

TERASOLUNA ViSC をある銀行システムの開発に利用した事例では、約 800ks の規模のソフトウェアを自動生成し、詳細設計から結合試験の期間を約 3 割短縮させることに成功している。

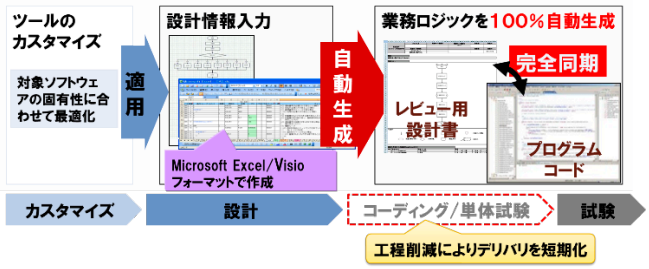

図 3：TERASOLUNA ViSC

テストの自動化技術

テスト作業は、テスト設計、テスト実装、テスト実行等、多岐にわたる。そのようなテストに要する様々な作業を自動化するための技術・ツールが開発・提供されている(図 4)。

また 2011 年にテスト自動化の知識体系として TABOK(Test Automation Body of Knowledge)が米国 ATI(Automated Testing Institute)より公開された。世界的にテスト自動化に関するノウハウの体系化が進んでいる。TABOK は特定のテストツールを対象とした知識体系ではなく、テスト自動化を考える上で必要な 12 のスキルカテゴリーを定義したものであり、自動化技術そのものというよりも利用ノウハウの重要性が認識されていることを示していると言える。

テスト作業	概要	主な自動化ツール
コードレビュー	ソースコードを査読し欠陥を検出する	静的解析ツール 構造解析ツール
テスト計画	テスト対象を把握し、テスト戦略・テスト条件を決める	トレーサビリティ管理ツール
テスト設計	テスト条件を漏れなく確認するためのテストケースを作成する	組合せテスト支援ツール モデルベースドテストツール
テスト実装	テストを実行するためのテストコードやテスト環境を作成する	テストコード生成ツール テストデータ生成ツール
テスト実行	テストケースに従いテストを実行し、欠陥の有無を確認する	テスト実行ツール（xUnit等） キャプチャ/リプレイツール

図 4：テスト作業と自動化

テスト実装のための自動化ツールとして NTT データ TERASOLUNA RACTES for UT(以降 TERASOLUNA RACTES)がある(図 5)。TERASOLUNA RACTES はテストケース表から JUnit テストコードを自動生成するツールである。具体的には、テストコード作成は、以下の

手順で行われる。

1. テスト対象のプログラムからテストケース表のひな型を自動生成する。
2. 生成されたテストケース表のひな型をもとにテストケースを記入する。
3. 記入済みのテストケース表からテストコードを自動生成する。

テストコードを作成することにより、保守フェーズにおける変更・修正に対して行うテストを大幅に自動化することができる。結果、保守フェーズのデリバリ時間は大きく短縮できる。ただ、作成しなければならないテストコードの規模は、テスト対象プログラムの 3 倍にもなると言われており、それをいかに効率的に作成するか大きな課題である。本ツールのようなテストコードの自動生成ツールによりテストコード作成の生産性を大きく向上することができる。

図 5：TERASOLUNA RACTES for UT

その他の領域の取り組み

8 つの自動化技術領域にうち、現在、NTT データが特に技術開発に力を入れているのが、「現行解析の自動化」である。老朽化したアプリケーションは保守コストの増大を引き起こしており、企業経営においても大きな課題となっている。長期間の保守により、アプリケーション構造はスパゲッティ化し、修正のたびに大規模なプログラム解析を要している。つまり、現行アプリケーションの理解に時間がかかり、その結果デリバリが遅くなる。

これに対して NTT データでは、稼働しているシステムのプログラムを自動解析し、用途に合わせた様々な設計書を正確に回復する技術 TERASOLUNA Reengineering を開発・提供している(図 6)。これにより現行システム仕様を短期間で正確に把握できるようになる。

また、このプログラムの解析技術と前述のプログラム自動生成ツール(TERASOLUNA ViSC 等)を組み合わせることにより、老朽化したアプリケーションから新しいアプリケーションを構築する「アプリケーションモダナイゼーション」をトータルに自動化することも可能となる。

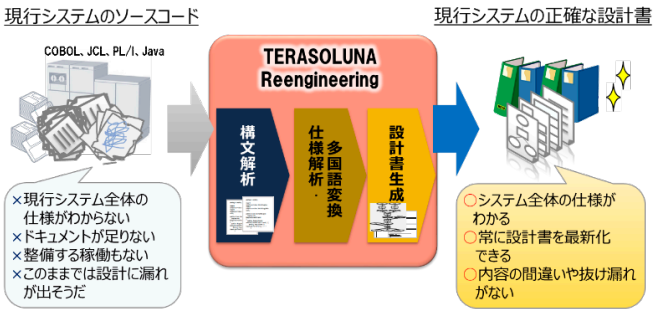

図 6：TERASOLUNA Reengineering

今後のビジョン

これまで説明したように、ソフトウェア開発の自動化の要素技術はすでに成熟化してきており、それらを開発全体でどのように活用して高速開発を実現するか、という適用ノウハウに重要性がシフトしている。今後は自動化の要素技術をもとにして開発全体を自動化する開発システムを仕立て上げることが、開発期間の劇的な短期化の肝となる。そしてそこにはどのようなツールをどのようなプロセス・手順で、どのような開発体制で実施するか、また自動化された開発に対してどのような品質管理を行うのかなどの多くの適用ノウハウが組み込まれたものになるだろう。

2013 年 3 月版