ZIPC Tester で加速する
NTT データのソフトウェア生産技術革新

ZIPC WATCHERS Vol. 17

株式会社 NTT データ 技術開発本部ソフトウェア工学推進センタ 課長
金子 武彦

はじめに

NTT データの生産技術革新への取り組み

NTT データは 2012 年度から 2015 年度にかけた中期経営計画における注力分野の一つとして，ソフトウェアの生産技術の革新を進めています．その目的は，従来の労働集約型の情報システム開発から脱却し，知識集約型に移行することによって，お客様により高付加価値なシステムを迅速に提供することです．

この生産技術革新を実現するための中核となる技術として，当社では「ソフトウェア開発自動化」の研究開発を行い，全社的な普及展開を進めています．ソフトウェア開発自動化とは，従来，開発者が人手で行っていたシステム開発業務の一部をコンピュータに行わせることを指します．その背景には，近年のソフトウェアやハードウェア，ネットワーク，データベースなどのコンピューティング技術の急速な進歩があります．それらを応用することによって，これまで人海戦術に頼らざるを得なかった開発作業をコンピュータで正確かつ迅速に処理できるようになり，システム開発における品質・コスト・納期のバランス関係に変革が起こります．例えば，システムの更改における現行システムの仕様の把握や，要件定義書や設計書などのドキュメント間の整合性の担保，設計書に従ったプログラムコードの製造や，テストケースやテストコードの作成と実施など，様々な開発工程を自動化することが可能です．

ただし，ソフトウェア開発自動化を導入した場合でも，お客様との要件・仕様の調整や，システムを取り巻く様々な制約や環境変化を考慮した設計など，本質的に人間でなければ実施できない開発作業は残ります．従来通りの高品質なシステム開発を担保しながらも，より創造的な作業に開発者が集中できるようにしていくことが，ソフトウェア開発自動化の本当の狙いです．

NTT データとキャッツの協業

ソフトウェア開発自動化のための研究開発の一環として，NTT データはキャッツと 2009 年の下期から協業し，状態遷移に基づいてテストシ

ナリオを編集・抽出するための自動化ツールの基盤を共同で開発しました．約 1 年半の開発期間を経て，その成果は両社のツールである ZIPC Tester と TERASOLUNA IDE ver.3 として結実しています．

両社の協業にはそれぞれ，相補的なメリットがありました．キャッツは 2009 年に NTT データグループに参画したことをきっかけに，組み込み分野に広く普及していた ZIPC の技術を情報システムの開発にも活用することを狙っていました．また，Java 技術の流行に伴い，ZIPC のツール群を Java 開発ツールのデファクト・スタンダードである Eclipse に移行させる必要性を感じていました．一方，NTT データでは情報システム分野において，開発成果物の自動生成を実現する「モデル駆動開発」の研究開発に長年取り組んでおり，Java や Eclipse を駆使して設計書やソースコードを自動生成するコア技術について強みを持っていました．この技術を現場のプロジェクトに普及展開させるために，実践的なツール開発に強みを持つキャッツと協業すれば，開発者の手になじむツールを開発しやすいと考えました．その結果，ZIPC Tester と TERASOLUNA IDE ver.3 はどちらも Eclipse ベースのツールとして実現されており，ZIPC の状態遷移モデルからテストケースを抽出するための基本的なエンジンや UI を共有しています．

本稿では，情報システム分野における生産技術革新の視点から，また，ZIPC Tester と共通の基盤を持ったツールを運用するユーザとしての視点から，NTT データのソフトウェア開発自動化の取り組みをご紹介し，テスト自動化の効果と可能性について議論したいと思います．

TERASOLUNA Suite

開発自動化ツールをワンパッケージに

「TERASOLUNA」は，NTT データにおけるオープンシステム向けのソフトウェア生産技術に与えられる統一的なブランド名です．ソフトウェア開発のための基盤となるアプリケーションフレームワークや開発・管理手順のほかに，開発生産性を向上させるためのツール群がラインナップされています．

当社では従来，設計，プログラミング，テストなどの開発工程ごとに多数の自動化ツールを用意していましたが，2014 年 1 月にそれらをシームレスにつなぐための開発自動化の総合的なソリューションとして「TERTASOLUNA Suite」をリリースしました．それ以前はツールを組み合わせて活用するための手順やノウハウが十分に体系化されていなかったために，開発自動化の効果を引き出すために現場のプロジェクトでツールの使い方を工夫していただいていることが多くありましたが，TERASOLUNA Suite は複数のツールを効果的に連携させるための開発手順や管理手順を含んでいます．現場でのベストプラクティスも積極的に取り入れており，今後も年 2 回程度のバージョンアップを通じて各ツールの機能改善やツール間の連携プロセスの見直しを継続していく予定です．

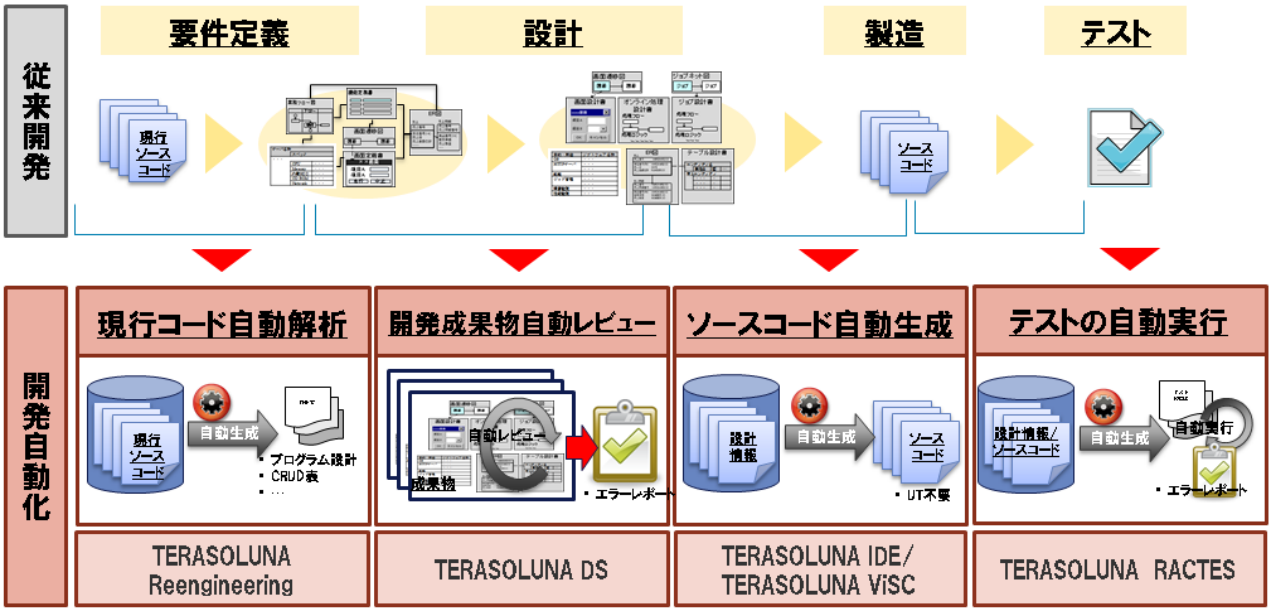

図 1 TERASOLUNA Suite の全体像

2014 年 10 月現在の TERASOLUNA Suite の最新版はバージョン 1.1 であり，TERASOLUNA Reengineering (以下 Reengineering), TERASOLOLUNA DS (以下 DS), TERASOLUNA IDE ver.3 (以下 IDE3), TERASOLUNA ViSC (以下 ViSC), TERASOLUNA RACTES (以下 RACTES)の 5 つの開発自動化ツールとそれらを活用するための開発手順，管理手順から構成されています．

図 1 はこれらのツールと対象とする開発工程の関連を示したものです．各ツールの大まかな役割は以下のとおりです．

- Reengineering はシステム更改の最上流の工程で活躍するツールであり，プログラム解析技術によってお客様の既存システムの資産を紐解き，設計情報を復元します．
- DS はドキュメントの情報を一元管理するためのツールであり，システム開発の過程で作成・更新される様々なドキュメントの情報を解析してデータベースで管理し，ルールに基づいて検証することにより，設計書間の整合性を担保します．
- IDE3 は対話型の情報システムの設計とプログラミングを自動化するツールであり，ユーザインタフェースにあたる画面遷移や画面の設計情報を入力することにより，設計情報の整合性のチェックや Java Web アプリケーションのためのソースコードや設定ファイルの自動生成を行います．
- ViSC は業務ロジックの設計書とソースコードの自動生成に特化したツールであり，Microsoft Excel の表形式で記述した業務ロジックやデータベースの設計情報を元に，設計書や Java ソースコード，設定ファイルの自動生成を実現します．

IDE3 と ViSC を組み合わせることによって，CSS, JavaScript などの画面の装飾やデザインに関する部分を除けば，Java Web アプリケー

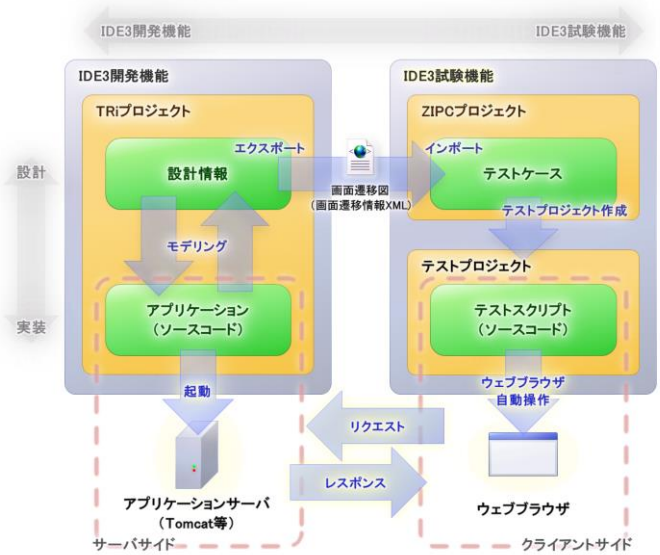

図 2 TERASOLUNA IDE ver.3 における開発機能と試験機能の関連

ションのソースコードを 100%自動生成することが可能です．さらに，Reengineering や DS を組み合わせることによってシステム要件の妥当性や設計の整合性を高めることが可能となり，TERASOLUNA Suite 全体で典型的なシステム開発プロセスの工期を大幅に削減します．

紙面の都合上，TERASOLUNA Suite を構成する各ツールをすべてご紹介することは難しいため，ここでは概要のご紹介に留めさせていただきます．本稿の残りの部分では，当社がキャッツと共同開発した IDE3 に話題を絞って，ZIPC Tester との共通部分や違いについて詳しくご説明します．

TERASOLUNA IDE ver.3

設計・プログラミング・テストを自動化

前述のとおり，IDE3 は Java Web アプリケーションの設計・プログラミング・テストを自動化するためのツールです．2013 年 5 月に最初のバージョンをリリースし，以降 5 回のマイナーバージョンアップを経て，2014 年 10 月現在は最新版として ver.3.5.0 を提供しています．リリースから約 1 年半で NTT データグループ内の 50 件以上のシステム開発プロジェクトに適用された実績を持っています．

IDE3 の機能は大きく分けて，Web アプリケーションの設計からプログラミングまでを自動化するための「開発機能」と，結合試験のテスト設計を支援し，テストコードの生成を自動化するための「試験機能」の 2 つに分かれます．

図 2 はこれら 2 つの機能における相互の関連を表したものです．

- 開発機能は開発者が入力した設計情報をもとに，実行可能な Java Web アプリケーションを生成します．
- 試験機能は開発機能から画面遷移，イベント，画面上の入出力項目などの設計情報を引き継ぎます．開発者は画面遷移のパスを元にテストケースを作成し，テストデータを与えることによって，テスト自動化フレームワークである Selenium に基づいた自動実行可能なテストコードを生成します．

開発者は開発機能を用いて生成した Java Web アプリケーションを Apache Tomcat などのアプリケーションサーバで実行させた上で，試験機能が生成した Selenium のテストコードをクライアントで実行します．この操作によって，

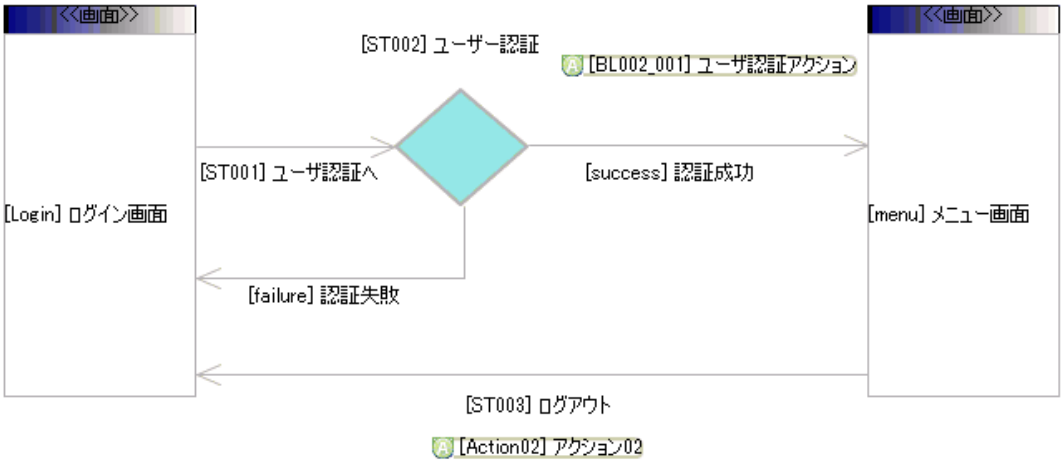

図 3 簡単な画面遷移図の例

クライアント上の Web ブラウザが自動的に操作されてサーバ上の Web アプリケーションにアクセスし，テストケースを自動的に実行することができます．

開発機能

開発機能では，開発者は Web アプリケーションのユーザインタフェースについて，主として以下の設計情報を入力します．

- 画面遷移: 画面をノード，画面間の遷移関係をエッジとした有向グラフ (画面遷移図) を記述します．図 3 は簡単な画面遷移図の例で，ログイン画面とメニュー画面およびそれらの間の画面遷移を含みます．

- 画面設計: ブラウザ画面内に表示されるテキストボックスやチェックボックス，ラジオボタン，プルダウンメニューなどの「入出力項目」とボタンやハイパーリンクなどの「クリック可能なイベント」を入力します．

- 入力値検証: 画面上の入力項目への入力値に関して，画面遷移の発生時に満たされるべき制約条件を入力します．また，制約が満たされない場合に表示するエラーメッセージを指定します．

- ビジネスロジック: 画面遷移のイベントが発生したさいに実行される処理を設計します．典型的なビジネスロジックは画面やデータベースとの間でデータを入出力したり，データの加工を行ったりします．開発者は処理の中でどの画面項目やテーブルからデータを取得し，どこに処理結果のデータを書き込むかを設計します．

- コードリスト: アプリケーションの業務ドメインに固有の概念について，ユーザが理解できるような「名前」と，それをコンピュータ上で扱うための「値」を設計します．例えば，性別のためのコードリストとして「男性」を 1，「女性」を 2 と指定します．

開発機能はこれらの設計情報を入力として，設計情報の整合性を確認し，納品物となる設計書やソースコードを自動生成します．IDE3 が生成するソースコードは，NTT データのシステム開発において豊富な実績を持つ Java Web アプリケーションフレームワークである TERASOLUNA Server Framework for Java (Web 版)に基づいたものであり，性能や信頼性

といった非機能的な側面においても高品質なシステムを構築することが可能です．

開発者が設計情報を編集するたびに，IDE3 はソースコードを随時，インクリメンタルに生成します．開発者は Java ソースコード，XML やプロパティなどの設定ファイルの記述箇所について意識する必要がなく，設計に集中しているだけでアプリケーションを常に実行可能な状態に保つことができます．これは，開発自動化ツールとしての IDE3 の大きな特徴です．

ただし，IDE3 は Java Web アプリケーションのソースコードのすべてを生成するわけではありません．例えば，画面の設計情報に対応して JSP (Java Server Pages) ファイルが生成されますが，ファイル内には入出力項目やボタン・リンクを動作させるための最低限のソースコードしか出力されません．画面のデザインやレイアウトについては JSP ファイルを手動で編集して完成させる必要があります．また，ビジネスロジックについても画面の表示確認を行うためのダミーデータを返却する最低限のコードを生成します．実際に必要となるビジネスロジックは開発者が手動でプログラミングするか，ViSC と連携して別途，自動生成する必要があります．

このように，設計・プログラミングの開発自動化は生産性を改善する上で大きな効果を持ちますが，高品質なシステム開発を実現するためにはテストも重要であり，大規模システムではテストの工数も膨大になります．当社では開発自動化によってテストの生産性も向上させたいと考え，IDE3 の開発機能と連携できるような試験機能を開発しました．

試験機能

試験機能では，開発機能から設計情報を引き継いでテストケースを設計し，テストを自動的に実施するためのプログラム (テストコード) を自動生成します．人手による従来のテストのやりかたと比較して，試験機能は以下の 2 点のメリットを提供します．

- 設計情報とテストケースの両方をツールが管理しているため，それらの整合性が保てます．人手でテストケースを設計した場合には，設計情報の解釈の誤りや見落としなどの人的なミスが原因で，テストケースにバグが埋め込まれる危険性があります．
- テストケースの実行を自動化できるため，テストの実施やデバッグに伴う再試験の工数を削減できます．人手でテストを実施する場合には，開発者によってテストのやり方が異なる場合にバグの発見や再現ができないことや，バグを修正した後の再試験にも開発工数がかかることが課題となります．

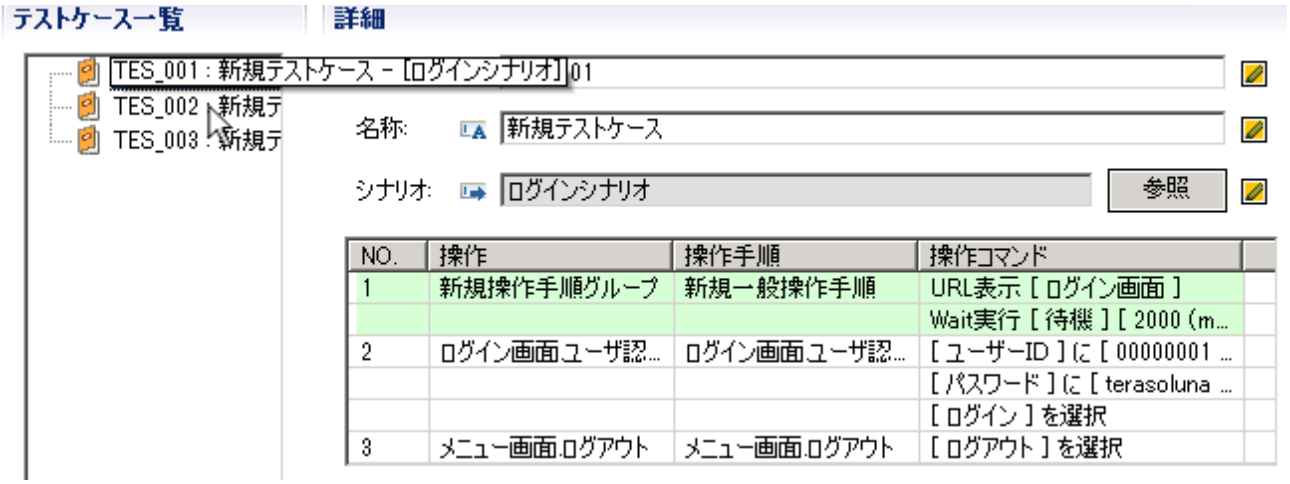

図 4 試験機能におけるテストケースの表示例

テストケースの設計には ZIPC Tester と共通の基盤を利用していますが，状態遷移モデルの記述には ZIPC を使用せずに，開発機能で作成した画面遷移図を状態遷移モデルとしてインポートします．このとき，画面遷移図上の画面を状態，画面遷移を状態遷移とみなし，画面上に配置されたボタンやリンクのクリックをイベント，画面遷移に伴って実行されるアクションやビジネスロジックを処理として解釈します．

試験機能を用いてテストケースを設計するためには，1) 基本シナリオの作成，2)　派生シナリオの作成, 3) シナリオへのテストデータの割り当て, の 3 ステップの作業が必要です．各ステップの作業について簡単にご説明します．

1. 開発者はまず，「基本シナリオ」と呼ばれる画面遷移の系列(パス)を指定します．

2. 次に，この基本シナリオをベースとして，条件に応じたシナリオを自動生成します．試験機能では ZIPC Tester と共通の経路抽出エンジンを採用しており，基本シナリオに含まれない画面遷移をたどる「寄り道」の回数や画面遷移の最大回数，必ず経由しなければならない画面などの条件を指定する事が可能です．生成されたシナリオを「派生シナリオ」と呼びます．開発者は基本シナリオに加えて，派生シナリオのうちテストケースで実施したいものを選択します．

3. 最後に，基本または派生シナリオに対して，各画面で入力するテストデータを割り当てます．シナリオに含まれる各画面でどのような入力項目が表示されるかをツールが把握しており，それらへの入力値を編集するためのエディタが表示されます．開発者はテキストボックスやラジオボタン，チェックボックス，プルダウンメニューなどへの値を指定するだけで，テストケースが完成します．なお，画面上に複数のボタンやリンクが表示される場合もありますが，どれをクリックするかは画面遷移に対応するイベントから自動的に決まりますので，開発者が入力する必要はありません．

例えば，図 3 の画面遷移図においてログイン画面からメニュー画面に遷移し，メニュー画面からログイン画面に戻る，という基本シナリオを考えてみましょう．この基本シナリオに対して

1 回だけ寄り道を許可した場合，「ログイン画面で入力値検証に一度失敗してから正常にログインする」，「認証処理に一度失敗してから正常にログインする」の 2 つのシナリオが派生します．テストデータとして，前者のシナリオでは必須入力であるユーザ ID かパスワードのいずれかを空欄とし，後者のシナリオに対してはユーザ ID とパスワードの誤った組み合わせを入力します．作成したテストケースの表示画面を図 4 に示します．また，テストケースの内容は Excel 形式のテストケース表として出力することが可能です．

IDE3 の試験機能と ZIPC Tester との大きな違いとして，試験機能ではテストの実行を自動化するためのテストコードを出力します．テストコードの生成のために開発者が行う作業はほとんどなく，テストで使用するデータベースの種類や，Web ブラウザの種類を選択するだけです．テストコードは Web アプリケーションのテストを自動化するためのフレームワークである Selenium を使用した Java プログラムとして生成されます．

このテストコードを実行すると Web ブラウザを自動的に起動して操作し，画面へのテストデータの入力やボタンのクリックを自動的に行って，テストケースの内容のとおりに画面遷移を自動的に実行します．このとき，テストケースの実施中の各段階におけるスクリーンショットを画像ファイルとして保存したり，ブラウザが表示している HTML やデータベースのテーブル情報をダンプしたりして，テストの実施証跡を記録することができます．

このようにテスト工程を大幅に自動化できるのが試験機能の特徴ですが，普及展開を進めるためにはツールの機能面だけでなく，利用者からのフィードバックに基づいた地道な品質改善も必要です．これまでにも，現場からの要望に基づいて Web ブラウザやデータベースなどの対応環境の拡充や，大量のテストデータの入力を効率化するためのユーザインタフェースの追加などを行いました．今後もこのようなフィードバックには継続的に対応していく予定です．

おわりに

さらなる生産技術の革新に向けて

本稿では，NTT データの生産技術革新に向けた取り組みである「ソフトウェア開発自動化」のの全体像として，様々なツールをシームレスに連携させるための「TERASOLUNA Suite」をご紹介しました．

また，ZIPC Tester と共通のツール基盤を応用した，Java Web アプリケーションの統合開発環境である「TERASOLUNA IDE ver.3」の開発機能と試験機能をご紹介しました．設計工程で ZIPC を用いていなくても独自の設計情報を流用して ZIPC Tester と同等のテスト設計が実現できることや，さらに一歩進んだ自動化のステップとしてテストコードの作成と実行までが自動化できることをお伝えしました．

TERASOLUNA Suite の普及展開はこの 1 年弱で始まったばかりであり，IDE3 の開発機能や試験機能も含めて，現場から寄せられた解決す

べき課題はまだ多くあります．NTT データではキャッツと連携し，今後も NTT データグループ全体に向けた生産技術の革新に取り組んでまいります．

ZIPC Tester を拡張したテストコード自動生成へ

ZIPC Tester は様々なユーザが使えるように汎用性を高めており，テストケースの設計やテストケース表の出力にフォーカスしたツールですが，キャッツでは ZIPC Tester のカスタマイズサービスも実施していると伺いました．

当社の社内ツールである IDE3 の事例のように，プログラミングやテストに適用する実装技術が限定できる場合には，カスタマイズによってテストコードの自動生成が実現可能でしょう．

読者の皆様も，自社の業務プロセスや利用技術応じたテスト自動化ツールを構築していただくことによって，テストの設計や実施に関する開発生産性を大きく改善できると考えています．ソフトウェア業界全体の生産技術の革新や底上げのために，本稿が参考事例として少しでもお役に立てれば幸甚です．