解 説 3

ツールによる効果的なシステム開発

ー よりスマートなシステム開発を目指して ー

世界がフラットになり、グローバル化が加速することで、企業を取り巻く環境はこれまで以上に急速に変化しています。その影響は瞬く間にわたしたちの社会全体に影響を及ぼすようになり、IT システムが経営に果たす役割はますます大きくなっています。市場の変化や競合他社の動向といった経営環境の変化をとらえ、いち早く経営に寄与するシステムを構築するために、システム開発プロジェクトには、これまで以上の高品質、低コスト、短納期が求められています。また、世界のフラット化とグローバル化の加速はシステムの開発現場においても同様です。世界中に分散された環境下でシステムを開発することが一般的となり、クラウド環境を活用した開発モデルにシフトするケースも増えています。

その一方で、現在の日本におけるシステム開発プロジェクトにはさまざまな課題点や問題点があります。この解説記事では、それらの課題点や問題点を解決して、お客様にとって競争力のあるシステムを効率的に開発するための IBM の取り組みについてご紹介します。

Article 3

Effective System Development Practice Using Tools

- Aiming at Smarter System Development -

Against a backdrop of accelerating globalization and an increasingly level global playing field, the business environment for enterprises has been changing faster than ever. The effects of these changes have had immediate impact on our whole society and the role of IT systems on business management is increasingly expanding. In order to develop systems that can quickly contribute to business management by capturing changes in the business environment in terms of market changes and competitors' trends, it is required that system development projects seek higher quality, lower cost, and shorter delivery time than ever. The global playing field is becoming increasingly more level and enterprises more globalized at an accelerated speed, and so the environment for system development is changing in step with this. Developing systems in a globally distributed environment has become a common practice and the number of system development cases employing a development model that uses cloud environment is increasing.

In the meantime, in today's Japan system development projects face a variety of problems and issues. In this article, we present IBM's work toward efficient development of competitive systems for customers by solving these problems and issues in their current system development projects in Japan.

❶ はじめに

1.1 日本のシステム開発プロジェクトにおける現状

企業を取り巻く環境の変化をとらえ、いち早く経営に寄与するシステムを構築するために、業務システム開発プロジェクトには、さらなる高品質、低コスト、短納期が求められています。その一方で、世の中の IT 関連雑誌や調査機関による調査結果を見ると、問題のあるプロジェクトでは、以下のような現状が見られます。

- 期待通りの品質を達成できなかったプロジェクトは、要件の未実装やテスト不足による品質上の問題が見られる。
- 予定コスト内で完了できなかった多くのプロジェクトは予算を増額し、その多くが追加開発作業につぎ込まれている。
- 予定スケジュール内で完了できなかったプロジェクトの遅延原因のトップは要件定義作業の遅延となっている。

システム開発の現場では、エンド・ユーザー部門、システム部門、ベンダー間において要求内容に関する認識のギャップがあり、要件定義に時間がかかるとともに、不十分な要件定義により、下流工程で大幅な手戻りが発生しコスト増の原因となったり、品質やパフォーマンスの問題がサービスイン直前になって発覚し、予定通りのスケジュールに対応できなかったりといった問題が発生しています。プロジェクト期間中に経営環境が変化することで、システムに求めるニーズ自体の変化も激しく、スコープ増加が多発することで、変更管理と計画変更作業に非常に多くの時間を費やすことになります。さらには度重なる要件変更に伴う設計作業の追加により、設計とソース・プログラムの整合性確保が煩雑となり、品質リスクを増大させています。

1.2 ソフトウェア開発のV字モデル

これらの諸問題を、ソフトウェア開発の V 字モデルに当てはめてみると、各局面において以下のような問題点が見えてきます（**図 1**）。

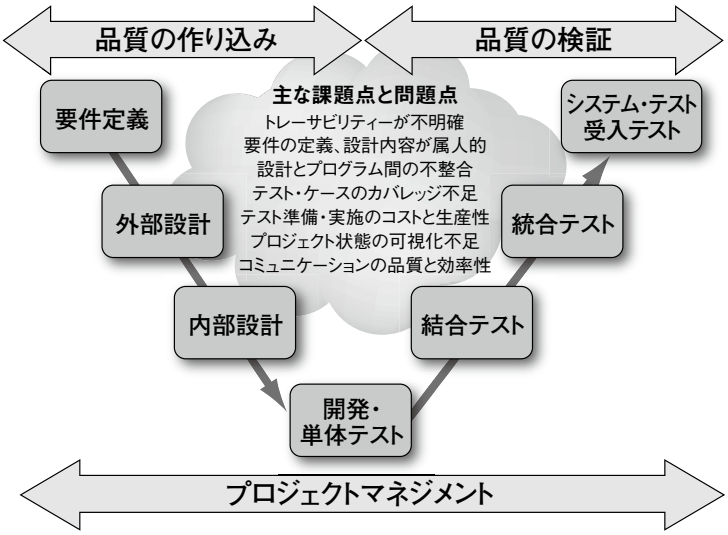


図1. ソフトウェア開発のV字モデルと問題点

(1)要件定義
- ビジネス要件とシステム要件の定義があいまい。
- 要件間のトレーサビリティーが不明確で影響分析が困難。
- それぞれの要件の定義において属人性が高く、ユーザーと開発者間で認識齟齬(そご)が発生。

(2)外部設計・内部設計
- ユーザー・インタフェース仕様が中心でビジネス・ルールの定義が属人的でユーザーが判断しづらい。
- システム要件とコンポーネント要件間のトレーサビリティーが不明確で影響分析が困難。
- 標準化が順守されず品質にむらがある。
- 設計書の一元管理や構成管理がされておらず、影響分析や保守が困難。

(3)開発・単体テスト
- 設計情報とプログラム仕様の間に不整合がある。
- 設計変更による影響分析漏れと修正漏れがある。
- テスト・ケースのカバレッジ不足。

(4)結合テスト・統合テスト
- バグ発生時に原因調査と影響分析に時間がかかる。
- 要件漏れや設計ミスによる手戻り工数影響大。
- テスト・ケース、テスト・データ作成の生産性と網羅性に課題がある。
- テスト結果のエビデンス取得と検証に時間がかかる。

(5)システム・テスト・ユーザー受入テスト
- システム・テスト、ユーザー受入テスト時点で仕様変更や追加開発が発生し、スケジュール遅延リスクが増大。
- 網羅性のある回帰テストに多くのコストが掛かる。
- 外部システムとの接続テスト環境の構築に多くのコストが掛かる。
- 外部システムとの接続テストの待ち時間が多く、柔軟に実施することが困難で、日程調整や事前事後作業に多くの時間とコストがかかる。
- システム障害時の問題解析に時間がかかる。
- 運用テストで非機能要件対応が不十分で甚大な対外的影響が発生。

(6)プロジェクトマネジメント
- 作業の進ちょく状況と遅延個所が正確かつタイムリーに把握できない。
- 品質の可視化ができない。
- 設計書のバージョン管理、プログラムのリリース管理、世代管理が正しく行えない。
- プロジェクト・メンバー間のコミュニケーションが効率的に実施できない。

これら、V字モデル上での課題解決に向けたIBMの取り組みについて、次章以降で解説します。

❷ ITによる経営への貢献とトレーサビリティー

2.1 要件管理の重要性

一般にプロジェクトの目標は「顧客の真のニーズを満たす高品質のシステムを期間内に予算内で提供すること」と表現することができます。しかしながら、米国 Standish Group [1] が行った調査 (CHAOS Report 2004) によると、プロジェクトの失敗要因として「ユーザーの要望を聞いていない」「要件や仕様が不完全」「要件や仕様の変更」といった要件にかかわる要因が上位に挙げられています。

要件管理とは一言で表現すれば、「システムに対する要件を引き出し、整理し、文書化する系統だったアプローチであり、システムの要件を変更する際に、顧客とプロジェクト・チームとの間の合意を形成し、維持するプロセス」といえます。このことからも要件管理がプロジェクトの成功において重要な役割を果たすといえるでしょう。

2.2 要件管理はなぜ難しいのか

要件管理の重要性は広く理解されているにもかかわらず、一方で「要件管理は難しい」「やってみたがうまくいかなかった」という声も根強く聞かれます。それはなぜでしょうか。前述の要件管理の定義にのっとり、要件の「引き出し」「整理」「文書化」「変更管理」の視点から要件管理を実践するに当たっての阻害要因が何であり、よく陥りがちな落とし穴が何であるかを整理します。続いてそれらをどのように打開すべきかを次節で考察します。

(1) 要件の引き出しにおける課題
(2) 要件の整理における課題
(3) 要件の文書化における課題
(4) 要件の変更管理における課題

（1）要件の引き出しにおける課題

要件管理を最も難しくしているのは、要件開発とも表現される「要件の引き出し」に関する課題ではないでしょうか。なぜ要件を引き出すのは難しいのでしょう。

その原因として、とりわけ近年のシステム開発において「真のニーズ」の把握が非常に困難になっていることが挙げられます。システムが経営に果たす役割が従来よりも大きくなり、市場の変化や競合他社の動向に応じてシステムに求めるニーズ自体の変化も激しくなってきています。これはまさに企業のビジョンや戦略に基づき、限られたIT投資を企業レベルでどこに優先的に振り向け、見込み利益・提供価値の予測を行いながらの投資判断を行うアプリケーション・ポートフォリオ管理の領域の必要性を示唆しています。実際には一旦予算化された後ではニーズの見直しは困難な現状がありますので、結果としてプロジェクト予算内でニーズの変化を吸収する努力の範囲を超えてしまった場合に下流になって追加投資の発生を余儀なくされるという事態につながっています。「真のニーズ」が従来よりも変化しやすくなっていることが、より要件の引き出しを難しくしています。

（2）要件の整理における課題

要件管理の概念が広まり、利害関係者（Stakeholder）から出てくる多岐にわたる要件を種類別に整理する必要性は広く知られるようになりました。ビジネス上の課題に対する利害関係者の望む状態を実現手段によらずに表現した「ニーズ（ビジネス要件）」、システムが外部に提供する観測可能なサービスや状態を表現した「基本要件（システム要件）」、システムが外部とどのように相互作用をするか、また非機能要件を表現した「ソフトウェア要件（コンポーネント要件）」のどれに当たるかを意識し、要件間の相互依存関係をトレーサビリティーとして表現する、というものです。

ところがいざ整理をしようとすると、このような要件の種類の分類に過度にこだわって時間を費やしてしまったり、大規模なプロジェクトで要件の数が非常に多いにもかかわらず人手で管理しようとして、膨大過ぎて管理できなくなり途中であきらめたりしまいがちです。また、ニーズを満たす基本要件がどこまで実現できているか測るすべがなく、基本要件がシステムにどう実装されているか影響分析できず、ビジネスに役立つシステムになっているか判断できない点も課題となっています。

（3）要件の文書化における課題

要件の記述は正確かつ明確で一貫性がありあいまいでない必要があります。要件管理の本質がユーザーとベンダーとの合意形成プロセスであることを考えると、文書化は非常に重要な作業となります。しかしながら、システムのユーザーとベンダーとの間の共通言語がいまだ確立されておらず、ニーズの表現方法の属人性が高く、結果としてユーザーのニーズをベンダーが正しく把握し切れない、あるいはユーザーがベンダーに正しく伝えにくいという状況が生じています。この状況の改善のためにBPMN（Business Process Modeling Notation）やユースケースといった手法も開発されていますが、これらの手段を用いずにあいまい性を排除しづらい日本語表記のみで要件をまとめようとしたり、要件を引き出すための手段としての利用よりもそれらの成果物を作ることが目的となったりしまいがちです。成果物を作ることよりも、ユーザーとベンダー双方が理解できるようにビジュアルに要件を見える化し、レビューやプロトタイプを活用して要件を引き出すことを重視しなければなりません。

（4）要件の変更管理における課題

ISO 9000:2005（JISQ9000:2006）[2]規格では、トレーサビリティーを「考慮の対象となっているものの履歴，適用又は所在を追跡できること」と定義しています。要件の変化に対応するために、トレーサビリティーを利用した影響分析と、要件が漏れなく下流の成果物に反映されているかというカバレージ分析の観点と、変更対応の抜け漏れ確認の妥当性検証が、要件の変更管理の主な目的となります。要件管理は限られた期間と予算内で実現できる要件の総量を劇的に増加させるものでは決してない点に注意が必要です。要件管理はベンダーの作業ではなく、あくまでユーザーとベンダーの合意形成を図り、真のニーズを実現する開発スコープを決定するためのものです。

2.3 課題の解消

要件管理は、究極的には限られた予算でより多くの要件を入れたいユーザー側と、変化は最小限にとどめたいベンダーとのせめぎ合いにほかなりません。このように人間系の作業が大半を占めるため、IBMでは要件管理の領域において、要件管理専用のツールを活用しています。なぜIBMではこの領域に注目しているのか。その理由は、この活動のほとんどがプロセス化やルール化がされておらず、手作業や属人的な進め方に頼っていることで、結果としてIT化の遅れが本来の経営競争力に影響を与えていると考えているからです。要件管理ツールを活用することで、要件を文章や表形式で表現でき、要件間のトレーサビリティーを定義することで、変更時の影響範囲と変更対応漏れの有無確認を簡単なツール操作で実施することができます。

一方、要件管理の課題にツールだけで対処するのではなく、要件のトレーサビリティーを一元管理し、影響分析や要件の抜け漏れの担保を属人性なく実施できることを重視したIBM標準の要件管理方法論も採用しています。このようにして要件の精度を高める努力を行っています。

最後に、日本情報システム・ユーザー協会が経営企画部

門 4,000 社を対象に行った調査（第 16 回企業 IT 動向調査 2010［3］）によると、ビジネス・プロセスの変革に対して IT 部門が期待に応えられている、もしくは同業他社より進んでいるというのは 51% にとどまっています。ビジネス・モデルの変革に至っては IT 部門が期待に応えられている、もしくは同業他社より進んでいるというのは 34% に過ぎません。すなわち IT が経営に寄与できているという昨今の経済状況下においてより限られた資本をより戦略的に投資したいと考えるのは当然であり、経営に寄与すると思えない IT の投資を絞ろうというのは当然のことだといえます。この章で繰り返し述べてきたように、昨今の経営を取り巻く変化の激しさに追随するため、システムに対する要件の変化も激しさを増しています。もはや要件が変更されていくことは前提とするべきでしょう。その上でベースラインを引き、変更のたびに影響を議論して受け入れと拒否を検討できる仕組み、その際にはビジネス目標や戦略の観点からと技術的な（アーキテクチャー）面からの双方から検討することが重要なのです。ニーズの変化に素早く対応して経営に寄与するシステムを作るため、もう一度要件管理に取り組んでみませんか。

❸ 設計仕様書のトレーサビリティーと、設計情報と整合性のあるコード生成

3.1 開発におけるトレーサビリティーの重要性と考え方

ソフトウェア開発におけるトレーサビリティーの目的は、ソフトウェア開発で管理される成果物やさまざまな事象を対象として、それらの間の関係を追跡可能にし、開発および保守における品質と効率を向上させることです。

成果物間のトレーサビリティーは、上流の成果物から、その成果物をインプットとして作られた下流の成果物への追跡できること、あるいは、下流の成果物から上流の成果物を追跡できることです。成果物間のトレーサビリティーは、開発工程に従った成果物作成作業の中で関連付けられ安定しているので、静的なトレーサビリティーということができます。

一方、作業・仕様変更、障害対応、テスト結果など成果物を修正する必要のある事象と成果物とのトレーサビリティーでは、5W1H（Why、What、Who、Where、When、How）といった情報やどのバージョンの成果物との関係なのかということも重要な情報です。例えば、障害報告があった場合、その障害の原因となったモジュール、ソースコードとの関係や変更要求に対するユースケースの関係などがあります。これらの事象と成果物間のトレーサビリティーは、開発工程全般において品質保証する観点でのトレーサビリティーとして動的に関係付けされるものであり、動的なトレーサビリティーということができます。

静的および動的なトレーサビリティーを確保することで、「要求が下流の成果物に漏れなく反映されているか検証することができる」「要求変更や設計変更が生じたときに影響範囲が把握できる」「問題の修正をソースコードにした場合、テスト範囲を特定できる」「追加開発時の見積もりに役立つ」などのメリットを得ることができます。トレーサビリティーを確保することは、品質保証の観点で大変重要ですが、これを人手で維持管理しようとすると大変な手間が掛かり、管理し切れず障害に至ることがあります。経験に基づく教訓から、トレーサビリティーに関する人の作業を軽減するツールを使うことは大変有効な手段となります。

3.2 アーキテクチャー管理とトレーサビリティー

ツールを活用しその効果を得る上では、物理的な成果物（ファイル）の関連を管理するだけではなく、論理的な設計要素（ファイルの中の特定要素）の関連を管理するレベルまでの実施を検討すべきです。そうでないと、例えば、物理ファイルの関連を ID で管理しても、結局ファイルを開いて中身を精査して影響範囲を調べるというような多くの工数がかかる作業を効率化することができないということになります。また、ソフトウェア構成管理によるベースライン管理を正しく行うことが前提であることも考慮すべき点です。

さて、ソフトウェアがますます複雑化する今日では、その枠組みの根幹をなす最適なアーキテクチャーの決定、さらにビジネス要求が正しくアーキテクチャーに反映されているかといった、フル・ライフサイクルを通じて整合性を確保することが極めて重要といえます。昨今、こうした、管理の視点でアーキテクチャーのトレーサビリティーを実施することがますます重視されていますが、その背景にある大きな要素としては、以下の 3 つが挙げられます。

1）開発したソフトウェア・コンポーネントやサービスを再利用可能にするためには、アーキテクチャーで定めた標準、プロトコル、インターフェースなどが、規定通りに実装されているかの統制が必要。
2）オフショアに代表されるソフトウェアの地域分散開発においては、要件、設計、実装のフル・ライフサイクルを通じてアーキテクチャーの整合性を確保することが非常に難しくなってきている。
3）アウトソーシングなどの契約形態では、要件定義と設計（アーキテクチャーを含む）までは社内、実装はアウトソース先などのケースが想定され、その場合、当初策定したアーキテクチャーがアウトソース先で正しく実装されているかの確認、整合性を取る必要性が高まってきている。

3.3 アーキテクチャー管理のための有効なツール

アーキテクチャーを一言で定義することは簡単ではありませんが、ここでは 1 つの視点として、プロジェクトの要件をサポートするためにプロジェクト・メンバーが従うべきガイドライン、決定事項、制約とします。

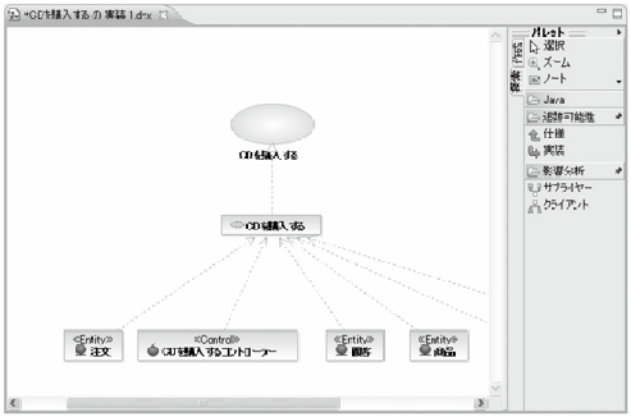


図2. RSAによる影響分析結果

IBMでは、アーキテクチャーをプロジェクト内外で共有するためにArchitecture Description Standardという標準を決め、表記法はUML（Unified Modeling Language：統一モデリング言語）を使うことを推奨しています。UMLを使う理由は2つあります。1つはデファクト・スタンダードを使うことによってコミュニケーション・コストを抑えること、もう1つはツールのサポートが得られることです。一般に開発ツールは、自動化による効率化、ヒューマン・エラーの低減、ガバナンスを強化、実現の支援を行いますが、アーキテクチャーの観点からは、プロジェクト・メンバーがガイドライン、決定事項、制約に従うことを支援するものだといえます。IBM Rational® Software Architect（以下、RSA）を例に紹介すると以下のような支援が提供されます。

- RSAが提供するパターンやモデル変換機能により、アーキテクチャー・パターンを自動化。
- UMLモデルのメトリクスとアーキテクチャー分析により、アーキテクチャー上好ましくないモデルの候補を察知。
- トレーサビリティーを利用した強力な影響分析は、アーキテクチャーとの関連を可視化し、変更があった場合の影響調査を支援（**図2**）。

詳しくは参考文献のWebページを参照ください［4］。

3.4 トレーサビリティーのチーム展開に有効なツール

ここまで説明したトレーサビリティーをプロジェクト全体で統制するには、管理のためのインフラを使用することが最も現実的です。近年、ALM（Application Lifecycle Management）をうたう開発のための管理基盤製品がさまざまなベンダーから提供されており、IBMではRational Team Concert™（以下、RTC）がそれに相当します。このような基盤製品を採用するメリットとしては、集中リポジトリー、柔和なガバナンス、リアルタイムな状況把握を挙げることができます。

(1)集中リポジトリー

作業と成果物（ソースコードやテスト結果）を履歴と共に関連付けて管理することで、動的なトレーサビリティーを実現できます。それらを保持するリポジトリーを単一にすることで、メンバーは容易に情報更新ができ、さらには情報のトレースも素早く行えます。

(2)柔和なガバナンス

プロジェクトの標準ガイドをメンバーに順守させる仕組みが管理基盤に備わっていることでトレーサビリティーを維持することができます。とりわけメンバー数がピークに達する実装工程やオフショア協業などでは徹底されないケースが増えます。RTCにはガイドを実装する機能があり、ユーザー操作がガイドを満たしていない場合には操作を完了させません。その際、ツール上にガイドが提示されるため自然とガバナンスが順守されます。

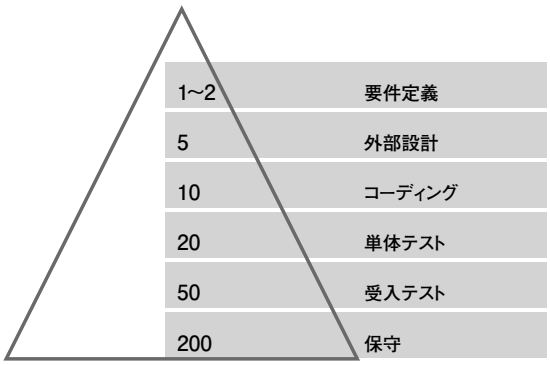
(3)リアルタイムな状況把握

副次的な効果としてチーム・メンバー全員がプロジェクトの進ちょくをグラフで視覚的に参照することが可能になります。RTCは標準でダッシュボード機能が稼働しており、誰でも多角的にプロジェクトの状況を把握できます。また、喫緊な状況において異なる担当者がコードの修正を行う場合であっても、常に正しいトレース情報を基に的確に作業を行うことが可能となります。

❹ テスト・ツールによる開発効率化

4.1 Full Lifecycle Testing（FLT）

ソフトウェアの信頼性と納期短縮が求められる環境で、IBMでは効率よく品質の良いものを作り出すためFull Lifecycle Testing（FLT）という考え方でテストを実施しています。FLTとは、ソフトウェア開発プロジェクトの早期段階も含めた全ライフサイクルにおいて、「テスト」を通じて品質向上を図るという考え方です。通常のシステム開発プロジェクトにおいて「テスト」というと、実行可能なモジュールを動作させるテスト（動的テスト）を指しますが、FLTではテストのアプローチとして、静的テストと動的テストの2種類を定義しています。静的テストとは、ドキュメントやコードについて詳細に検査するテストのことで、具体的な実施方法として、「計画レビュー」「要件のウォークスルーと承認」「設計およびコード・インスペクション（検査）」「テスト・ケース・レビュー」などが挙げられます。一方の動的テストとは、いわゆる実行可能なモジュールを実行し、その動作を検証するテストのことで、具体的な実施方法として、「アプリケーション・プロトタイプによる検証」「稼働システムに対するテスト・ケースの実行」「エンド・ユーザーとの利用シナリオの実施による使い勝手の確認」



出典:Robert B. Grady(1989)著書を元に作成

図3. 欠陥を検出する工程と修正コストの関係

「並行稼働による本番環境のテスト」などが挙げられます。

Robert B. Grady（1989）によると、運用時に見つかる欠陥の修正コストは、要件定義時の200倍掛かると言われています（**図3**）。そのほかにも類似の調査結果があり、ソフトウェア・テストの世界では広く知られていることです。

このようにプロジェクトの早い段階で問題を見つけて解決することは、品質と効率の両方を満たすために有効となります。そのためには、プロジェクトの初期段階から静的テストも含めたプロジェクトの全ライフサイクルにわたってのテスト計画を立てることが重要です。また、計画されたそれぞれのテストにおいて、「計画」→「準備」→「テスト実施」→「レポート」というステップでテスト・プロセスを管理していくことも重要になります。

FLTでは、テスト計画の立案をさらに効率化するために、テスト・ツールの活用を推奨しています。一般にテスト計画を立てる場合、品質（Quality）、コスト（Cost）、納期（Delivery）のトレードオフ関係にある3つの要素を満たす必要がありますが、すべてを網羅する完璧なテストは実質不可能です。そこでテスト・ツールの特性を生かして、テストのある部分をツール化することで、人手だけでやったときより良い品質、コスト、納期の組み合わせに落とし込むことができるのです。また、ツールを使うことで動的なトレーサビリティーを確保しやすくなります。以下では、FLTでいう静的テストで使う静的テスト・ツールと動的テストで使う動的テスト・ツールの2つに分類して説明します。

4.2 静的テスト・ツール

静的テストのツールとしては、コード解析ツールやモデル解析ツールなどがあります。コード解析ツールは、ソースコードを対象として、標準のコーディング規約を順守できているか、メモリー・リークなどの潜在的バグや性能劣化の原因となる部分がないかなどを検出するツールです。モデル解析ツールは、UMLでモデリングされたモデルが前もって決められたルールを順守できているか、クラスの参照が集中していないかなどを検出するツールです。これらツールの対象となっているコンパイル済みのソースコードやモデリング・ツール内で管理されるUMLモデルは、それ自体がソフトウェアによって扱いやすく、比較的に昔からツール化が進んでいる分野です。しかし、そのほかのMicrosoft® WordやMicrosoft Excel®に代表されるオフィス・アプリケーションで作成されたドキュメント類については、一般的に人の目視によるインスペクションによって静的テストが実施されてきました。人の目視によるインスペクションでは、インスペクションする人のスキルにバラツキが大きい上、大きなプロジェクトでは膨大なドキュメントを網羅することが難しくなります。この分野でツール化が進まない原因としては、オフィス・アプリケーションで作成されたドキュメント類からテストに必要な情報を取り出すことが難しく、またプロジェクトによってさまざまな種類のフォーマットのドキュメントが存在しているからです。

このように従来はツールによる静的テストの対象になり難しかったオフィス・アプリケーションのドキュメント類ですが、それを可能にしたツールが、IBM東京基礎研究所が開発した「ドキュメント品質検証ツール」です（本誌78ページ以下：IBM Research テクノロジー最前線参照）。「ドキュメント品質検証ツール」を使うことで、大量のドキュメントに対する全体的な分析と個別ドキュメントのニーズに対応した分析が可能になり、これまで人手で検証していたコストを大幅に削減することができます。

4.3 動的テスト・ツール

一方の動的テストのツールとしては、個別のクラスや関数などを対象にテストの実行、コードのカバレッジ（C0、C1、C2など）の計測、テスト・データやスタブの作成などの機能を持った単体テスト・ツール、実行時間やメモリー使用量を監視し、性能上のボトルネックやメモリー・リークを発見するためのプロファイル・ツール、実際に動くプログラムを対象に機能要件および非機能要件（セキュリティー、パフォーマンス、プライバシー・ポリシーなど）をテストするツールなどがあります。このうち機能要件を確認するテスト・ツールとしては、各種プロトコルや、人との間を仲介するさまざまなインターフェースをシミュレートするテスト・ツールが存在します。その中で人とのインターフェースをシミュレートするテスト・ツール、いわゆるテスト自動化ツールが、製品やフリー・ソフトウェアとして多く存在しています。これらの人をシミュレートするテスト・ツールの多くは、対象となるアプリケーションを実際に人が操作して、操作手順を記録し、ツール上で再生することで人の操作をシミュレートします。こういった記録・再生型のテスト自動化ツールは、直感的に自動化を始めることができるのですが、プロジェクトの中で使うと幾つかの問題が発生することがあります。1つ目は、画面の仕様が変更すると以前に記録したスクリプトが使えなくなり、再度記録し直さなければならないこと、2つ目は動くプログラムができるまで記録できないので、早期にテ

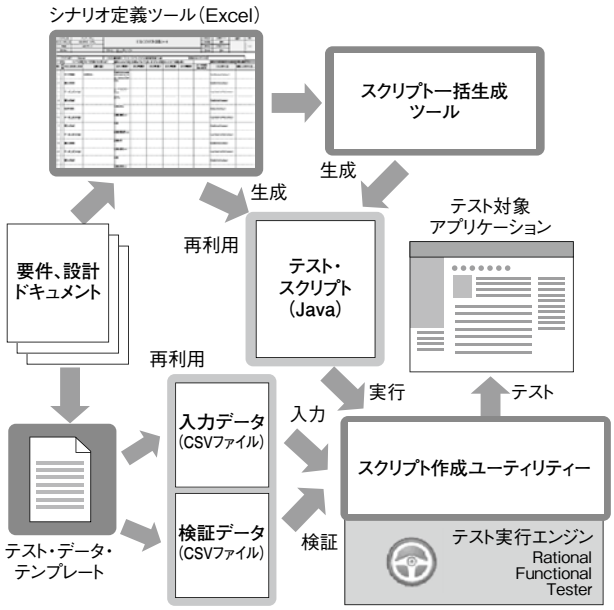


図4. GTAM Test Tool

スト準備ができないことです。そして3つ目は、テスト・シナリオごとに最初から最後まで記録するため、シナリオの再利用がやりにくいということです。

以上のようなプロジェクト現場の声を反映し、より効果的に使うことを目的に開発されたIBMの利用可能ソフトウェア資産がGuide and Toolkit for Application Modeling（以下、GTAM）のテスト・ツールです（**図4**）。GTAMは、IBMのテスト自動ツール製品であるRational Functional Tester（RFT）をRuntimeのテスト実行エンジンとして動作させます。操作対象のコントロール（ボタンや文字入力フィールドなど）に対する識別の仕方の違いと操作の記録を行わず、代わりに操作方法（テスト・シナリオ）を直接Java ™やシナリオ定義ツールというExcelベースのツールを使って記述していきます。また、操作対象のコントロールを認識するためにIDやNameといった操作対象のコントロールに一意に付けられた識別子を使います。そのため、テストを意識して設計の段階からそれぞれのコントロールに対し一意になるようにIDやNameを決めてもらう必要があります。IDやNameだけとはいえ、一意になるように管理しなければならないので、プロジェクトにとっては負担になりますが、それ以上に多くのメリットがあるのです。

1つ目のメリットは、一意なIDやNameで識別しているため、操作対象のコントロールのIDやNameが変わらない限り、同じスクリプトを使って自動実行することが可能になるということです。2つ目は、操作を記録するのではなく、IDやNameを元にテスト・シナリオを直接JavaやExcelで記述するため、アプリケーションが動く前でも自動テストの準備を始めることが可能になること、そして3つ目は、コンポーネント化しやすいということです。

一般的に記録・再生式の自動テストでは、始めから終わりまでの長いテスト・スクリプトが生成されるため、コンポーネント化しにくいのですが、IBMソフトウェア資産の場合、1つの画面遷移のテストを1つの単体テスト・スクリプトとすることにより、後続の統合テストやシステム・テストにおいて、単体テスト・スクリプトを部品として組み合わせたスクリプトを作ることが可能になります。コンポーネント化したテスト・シナリオを再利用することで、テスト効率化と品質向上が期待できます。

「画面変更に強い」「早期にテスト準備が可能」というメリットは、V字モデル（**図1**）の左側の仕様を右側で検証するというつながりを強化し、「コンポーネント化」というメリットは、右側の単体のテストから統合テスト、システム・テストへのつながりを強化します。

これらの特長を生かし、実際のプロジェクトに適用している例として、日本生命保険相互会社様の「新統合システム新事務（新契）工程管理システム構築プロジェクト（以下、新統合計画）」があります。新統合計画は、保険の提案や引き受けから保険金・給付金などの支払いに至るまでのあらゆる仕組みを、分かりやすさや利便性の向上といった観点から見直すプロジェクトです。開発するアプリケーションは、一般的なWebアプリケーションなのですが、保険業務特有の画面が複雑で項目数が多く、1つの作業のオペレーションが長いのが特徴です。そのため、人手でテストを行った場合の誤りや作業の効率化を考慮し、当初より自動化テストが考えられていました。しかし、記録・再生式の自動化テスト・ツールを使った場合、画面変更のたびに記録し直さなければならず、思ったほど効率化できなかったという経験があったことから、画面変更に強いGTAMを採用しました。

現在、よりプロジェクトに合った仕様にするためのカスタマイズと並行してテスト作業を進めています。今後は、共通のフレームワークを採用しているほかのシステム構築プロジェクトにおいても、採用が予定されています。

❺ アプリケーション仮想化によるテストの効率化

5.1 アプリケーションの仮想化とは

ここまで、各局面におけるトレーサビリティーの確保とテスト実施の効率化について解説してきました。テストの終盤では、ユーザーのビジネス要件およびシステム要件が正しくシステムに実装されていることを検証することになります。テスト環境や検証内容も本番システムを想定したものになるので、効率的な実施が大きな課題となります。近年、IBMが注目するアプリケーションの仮想化技術は、アプリケーション領域までを仮想化することで、テスト環境構成そのものをスリム化し、さらなるコスト削減を追求するためのものです。

5.2 テストにおける課題

アプリケーションの仮想化はテストにおいてその効果を発揮します。アプリケーション開発の終盤の作業はテストが大半を占め、プロジェクト全体のコストの約 3 割をテストが占めているともいわれています。このテストにおける大きな負担は、テスト環境の構築費用とテスト環境の利用に関する待ち時間です。

複数のテストを並行して効率よく実施するためにはそれだけ多くのテスト面を必要とします。そのテスト面の中には、テスト対象のアプリケーションを動作させるために必要なテスト対象外のアプリケーションも含める必要があります。このようなテスト対象以外のシステムへの投資が大きな負担となるのです。この負担にはハードウェアの購入費用、固定資産の減価償却、ソフトウェア・ライセンスの購入費用、ソフトウェア・ライセンスの保守費用などが含まれます。規模の大きい開発事業では、この負担は非常に大きなものとなります。

一方で、こういった負担を軽減するために、テストで利用できる資産をプロジェクト内で共用にしたり、ほかのプロジェクトでも利用されている既存の資産を流用したりするという対処方法がありますが、この場合は、各テストで利用できるシステムの利用時間が限られるため、テストの生産性に影響が生じます。このため、プロジェクト期間の長期化の原因となりコストの発生源となります。

さらにこういった課題を回避するためにシミュレーターなどの開発が試みられるのですが、予想以上に工数がかかったり、投資を行っても対象システムを忠実に再現するものが作れなかったりします。

アプリケーションの仮想化はこういった課題の解決を目指したものです。テストに必要な周辺環境を、できるだけコストを掛けずに容易に構築できるようにするための技術です。

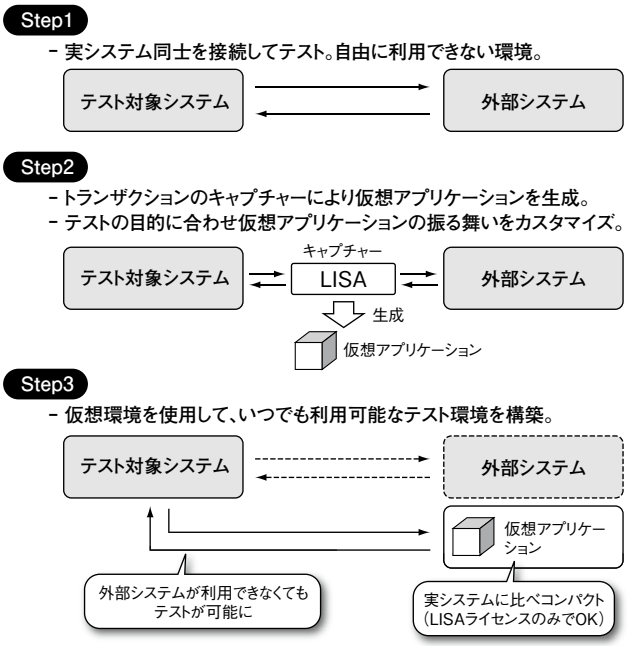


図5. LISAを用いたアプリケーション仮想化

5.3 LISAによるアプリケーション仮想化

IBM では iTKO 社の LISA という製品技術を活用することでこの課題を解決しようとしています。LISA はテストにおける外部システムへのトランザクションを記録し、それを忠実に再現する能力を持っています。LISA は HTTP、MQ、JMS、JDBC などのプロトコル上で交わされるトランザクションをキャプチャーし、実システム同等の応答を再生できる仮想アプリケーションのモデルを生成することができます。このモデルを仮想アプリケーションの実行環境に展開して動作させることで、あたかも接続先のシステムが稼働しているかのようにテストを実施することができます。LISA を用いた仮想化のイメージを**図 5** に示します。

このように LISA を利用することで、これまで実機で行ってきたテストを、仮想アプリケーションを利用した環境に容易に移行することができます。

5.4 LISAを利用することの利点

ここで、どのような環境でアプリケーション仮想化が効果を発揮できるのか、より効果的な仮想化が検討できるように、アプリケーション仮想化を実施することで得られる効果にはどのようなものがあるのかについて整理してみたいと思います。アプリケーション仮想化による効果は、テスト環境に掛かるコストの削減、テストにおける待ち時間の解消、テスト作業の効率化、テスト作業の作業効率化として現れます。このような効果を以下のような LISA の特長により生み出されています。

① LISAによるシミュレーション環境は、安価なLinux®あるいはWindows®マシン構成で実現できLISA以外のソフトウェアを必要としない。
② 仮想化されたアプリケーションでは、テストのための準備（テスト・データ投入、テスト前状態システムのバックアップ）、テスト後の事後作業（作成データの削除やシステム環境のリストア）が不要である。
③ LISAはキャプチャーによりシミュレーターを開発することが可能で、自作する場合よりも効率的にシミュレート環境を構築できる。

(1) テスト環境

冒頭で述べたようにテスト環境を構成するためには、テスト対象アプリケーションが処理を実行する際にアクセスするシステムを用意する必要があります。ここに掛かる費用については、削減を検討することができます。並行テストを可能にするために、必要なテスト環境分すべてに用意するという方法

ではなく、実機が必要な場合とそうでない場合に分けて、一部を仮想アプリケーションに置き換えることで費用の削減を検討することができるのです。

(2)テストの待ち時間解消

複数のチームが並行してテストを行うことをあきらめ、テスト対象外の外部システム用のハードウェアおよびソフトウェアの費用を抑えてテスト環境を構築した場合には、テスターのテスト環境利用時間に制約がでます。アプリケーション仮想化はこういったテスト環境をシェアして利用する場合の時間的制約を緩和することに役立てることができます。実機での接続検証が可能な場合に限定して、実機テストを行い、それ以外の用途には、仮想化されたアプリケーションを利用することで、実機では費用面であきらめることになった、占有のテスト環境を各チームに提供することが可能になり、生産性と品質向上を期待することができます。

(3)テスト作業の効率化

テストの実施においては、その事前準備および事後処理の時間も想定しておかなければなりません。テスト対象の接続先として実環境を使う場合には、事前準備の中には、利用のための事前調整、テスト・データの事前作成やテスト後の現状復帰のためのデータ削除やリストア作業などが含まれます。仮想化されたアプリケーションを利用したテストの場合には、これらの作業が不要となるため、実機を必要としない場合に適用することで、こういった準備工数を削減することができます。またLISAはシミュレーターを開発するためのツールとして非常に優れた特長を持っています。独自に外部システムなどのシミュレーターを作成するよりも高いROIが期待できます。

5.5 クラウド環境の活用

アプリケーション仮想化という考え方は、開発サイクルの短縮、テスト環境に掛かるコストの削減に大きく貢献するソリューションであると思います。クラウド環境を活用することで、十分なテストが行える環境が安価にかつスピーディーに提供できるようになることが期待されるため、今後積極的に展開していきたいと考えております。なおLISAの適用事例については、Webサイト[5]に事例紹介の記事があります。

5.6 効果的に取り入れるために

今回紹介したようなテスト・ツールを活用してプロジェクトの生産性を上げるためには幾つかのポイントがあると考えています。

(1)テスト計画の重視

1つはどの領域に適用するのかの検討をテスト計画の中で議論をして、効果が得られる領域を決めるというテスト計画の作業です。ほかのツール技術同様にいえることなのですが、IBMで推進するFLTではテストにおけるこのような計画の作業を重視しています。

(2)テスト専門チーム

2つ目は、このような専門的なツール技術を導入してテストを実施する際の推進体制です。近年テストの分野ではさまざまな自動化や効率化が進んでおり、すべての技術者がこれらのツール技術に精通して、効率的に開発作業を行うことが難しくなってきています。同時並行する複数のプロジェクトそれぞれで、ツールの標準を決め、技術者を教育することの負担が大きいため、プロジェクトを横断しての推進体制を構築することが有効です。

❻ まとめ

今回ご紹介しましたように、IBMではさまざまなツールを活用してプロジェクトを実施していますが、ツールを活用すれば高品質で効率的なプロジェクトができるとは限りません。IBMでは、基本となる開発方法論と、当解説でご紹介した要件管理の方法論やFLTといったテストの考え方を組み合わせ、それらの方法論をより効果的に実践するためにさまざまなツールを活用してシステム開発プロジェクトを推進しています。適切なツールを効果的に活用することで、各要件とトレーサビリティーを管理し、経営に寄与する高品質なシステム開発の実現に取り組んでいます。

[参考文献]

[1] Standish Group, http://www.standishgroup.com/ (2004).
[2] 国際標準化機構 :ISO 9000:2005 品質マネジメントシステム―基本及び用語, (2005).
[3] 社団法人 日本情報システム・ユーザー協会 :第16回企業IT動向調査2010,http://www.juas.or.jp/servey/it10/press-pp100409.pdf
[4] IBM Rational Software Architectのここが凄い,http://www.ibm.com/software/jp/rational/products/design/rsa/highlights/
[5] developerWorks® 電力業界のためのサービスの仮想化と検証のプラクティス, http://www.ibm.com/developerworks/jp/cloud/library/cl-servicevirt/

日本アイ・ビー・エム株式会社
グローバル・ビジネス・サービス事業
アプリケーション開発事業
テスティングサービス・テストエンジニアリング
マネージャー、シニア ITA

石橋 正章 **Masaaki Ishibashi**

[プロフィール]
入社以来ソフトウェア製品のテストを担当。特に自動化による効率化を中心に活動。2007 年に現所属であるテスティングサービスに異動、テストの効率化および第三者テストの普及活動に向け活動中。

日本アイ・ビー・エム株式会社
グローバル・ビジネス・サービス事業
アプリケーション開発事業
メソッド&ツール担当
シニア IT アーキテクト

寶生 一成 **Kazunari Hohshoh**

[プロフィール]
1991 年、日本 IBM 入社。93 年よりサービス部門にてお客様の業務システム開発プロジェクトに従事。2006 年より社内向けの技術支援サービスやプライベート・クラウド（GaIN）の展開を担当し、2011 年から開発標準のメソッドとツールの展開活動を担当。

日本アイ・ビー・エム株式会社
ソフトウェア事業ラショナル事業部
クライアントテクニカルプロフェッショナル
主任 IT スペシャリスト

大澤 浩二 **Kohji Ohsawa**

[プロフィール]
2000 年日本 IBM 入社。モデル駆動型開発による大規模アプリケーション構築プロジェクトに従事後、オブジェクト指向分析設計のための IBM 知的資産の開発を担当。2009 年より IBM Rational 事業部に異動、現職。要求管理およびオブジェクト指向分析設計の普及活動に従事。

日本アイ・ビー・エム株式会社
ソフトウェア事業ラショナル事業部
クライアントテクニカルプロフェッショナル
アドバイザリー IT スペシャリスト

熱海 英樹 **Hideki Atsumi**

[プロフィール]
一貫してソフトウェア開発市場に従事する。近年は開発支援ツールのプリセールス・エンジニアを務め、2010 年 12 月に日本 IBM 入社。現在は、主に Rational Team Concert のプリセールスを担当。

日本アイ・ビー・エム株式会社
グローバル・ビジネス・サービス事業
アプリケーション開発事業
EA&T. ハイ・バリュー・ソリューション担当
シニア IT アーキテクト

小倉 弘敬 **Hiroyuki Ogura**

[プロフィール]
1987 年、日本 IBM 入社。大和研究所にて OS/2 やオフィス・アプリケーションの製品開発を従事。2009 年よりグローバル・ビジネス・サービスに移り、モデル駆動開発やテスト自動化のための IBM 資産である Guide and Toolkit for Application Modeling の開発、デリバリーをリード。