解説 3

テスト仮想化技術による生産性向上と期間短縮の実現

― テスト実施に対する生産性阻害因子の分析とその対処法 ―

われわれを取り巻くアプリケーション・システムの複雑性は、ソフトウェア技術の進化とともに急激に増加しています。2010 年に発表された「IBM Global CEO Study」では、多くの CEO がシステムの複雑さが今後さらに増加すると予期しながらも、その軽減策や対応計画などの準備ができていないと答えています。そのような複雑なソフトウェア開発のテスト工程において、複雑性や不規則性がどのようにテストの生産性阻害因子となり得るか分析しました。本稿では現在の開発現場にて日常的に行われているテスト工程の生産性を阻害する代表的な項目を 3 つ取り上げ、その概要と解決策についてテスト仮想化技術がもたらす新しい手法と併せて解説いたします。テスト仮想化とは、いわゆる昨今のシステム資源の仮想化とは異なり、結合・接続テストにおける相手先の「仮想化」を意味しており、テストの期間短縮、生産性向上およびリスク低減を実現する技術として注目されています。

❶ ソフトウェアやシステムの「不規則性」と「複雑性」

現代の混沌としたシステム開発や保守の現状は、さまざまな科学分野における発展の道程に追従し、飽和点に達したといわれています。単一の企業が保有するシステムや 1 つの工業製品が有するコード量、コンポーネント数（すなわち、システムの構成要素数）、要素技術の数などは「不連続」「不規則」「非線形」の様相で増大し、コンピューターの信頼性・拡張性を将来にわたって考慮することが極めて困難な時代を迎えました。

システムやソフトウェアの「不規則性」について例を挙げると、システム改変の頻度は企業や組織の予算やリリース・プロセスに依存していることから、周期性を有して一定であるといわれていますが、その改変量や改変の難易度は必ずしも一定ではありません。これはソフトウェア製品開発における派生開発においても同様です。その理由は、毎回受け取るソフトウェア機能改変要件の複雑さが一定ではなく、さらにソフトウェアやシステムの複雑度は年を経るごとに非線形で増大するからであると考えられています。

1974 年に発表された Lehman's Law（リーマン法則）[1] では、以下の 6 つの法則が定義されています。

第 1 法則：Continuing Change
第 2 法則：Increasing Complexity
第 3 法則：Conservation of Familiarity
第 4 法則：Continuing Growth
第 5 法則：Declining Quality
第 6 法則：Feedback Systems

そのうち、注目すべきは 2 番目の法則であり、次のように定義されています。

「第 2 法則：特別な軽減策を施さない限り、システムの複雑さは増大し続ける」

当時から現代に至るまで、複雑性の増大について、ソフトウェア・システム保守の分野でさまざまな警鐘が鳴らされていますが、それにもかかわらず、「不規則性・不連続性」に対処し、複雑度を低下させる取り組みはなぜこんなにもおろそかにされてきたのでしょうか。

また、2010 年に発表された「IBM Global CEO Study 2010」[2] では、8 割の CEO が今後のビジネスを支えるシステムの複雑さはますます増加すると予期していますが、その複雑化するシステム環境に対して十分な備えがあると答えたのは全体の 5 割にとどまっています。

本稿では、ソフトウェアのテストを取り上げ、複雑性や不規則性について考察し、その対処技術である「テスト仮想化」について言及していきます。

❷ テスト工程の生産性阻害要因とは

皆さまの組織で行われるソフトウェア・テストにおいて、最も工数を要するテスト・レベルはどこにあるでしょうか。多人数の参画を必要とするユニット・テスト・レベル、あるいはリリース直前のシステム・テスト・レベルなどが考えられます。

以下に、工数を要する代表的なソフトウェア・テストとその理由を列挙し考察します。これら所要工数が大きい理由は、前述の「複雑度」の考慮や管理が十分でない場合が多いことが想定されます。

2.1 結合テストで「待ち」が発生している

結合テストは個々のコンポーネントのユニット・テスト完了後、すなわちコンポーネント・アップの状態を待って開始されます。そのため全体のスケジュールは、ユニット・テスト完了が最も遅いコンポーネントの進捗に影響を受けることになります。

これは、古くから「ビッグバン・テスト」といわれているテスト・スタイル、つまりは全モジュールを一気に完成させ、全部一度に結合して「トランザクション」を一斉に動作検証するスタイルの結合テストと同じであるといえます。

必ずしも、ビッグバン・テストが「悪い」わけではないですが、一般的に次のように品質確保する上で難易度が高くなるといわれています。

1）複数欠陥が同時に取り込まれるため、問題判別の難易度が高くなる。
2）テスト実施者にとって、多くの認識できない事象を含み、テスト管理の難易度が高くなる（=高リスク環境）。
3）最遅コンポーネントの開発に引きずられ、チーム間で「完成待ち」が発生し、コストがかさむ。

一部の小規模なシステム開発を除き、一般的にビッグバン・テストは、通常のトップダウンやボトムアップ・アプローチによるテスト管理よりも、混乱をきたす確率が高いでしょう。

結合テストでの複雑度の軽減には、幾つかの技法が存在します。ビッグバン・テストに代表される「一斉テスト」は、その対象コンポーネント数や開発者およびチーム数が増大すればするほど複雑度が増加し、失敗する確率が高くなります。従って、通常のトップダウン、ボトムアップ・アプローチと同様に、小さい単位で部分的に結合させて段階的につなげていくテスト形態が、複雑度軽減の観点からは望ましくなります。この考え方は「増分結合テスト（インクリメンタル・インテグレーション・テスト）」と呼ばれています。

前述 1）～ 3）は「1 回の結合工程ですべてを検証する」ことに起因する難易度上昇であり、それは取りも直さず分割統治設計と段階的結合（=増分結合テスト）によって対処可能なものです。つまり、部分結合テストを可能にする技術の必要性が生じているといえます。

2.2 対外システムへの接続テストには多大な準備工数が掛かる

技術要素の組み合わせ数は年々増大し、システムを構成する機能は追加に次ぐ追加で肥大化しています。複雑性が増大する昨今のシステム開発においては、プロジェクト全体進捗を阻害しない結合テストを小規模および部分的に実施できる仕組み、あるいは段階的な結合／接続試験を実施できる仕組みが望まれています。

例えば、社外やネットワーク・リソースに対する接続テストなどのシステム・テストにおいても同様です。

対外接続を伴うシステム・テストは、多大な準備工数を要する場合が多く、また将来自社システムと接続の可能性があるサービスやシステムの数は正確に予想・予測が不可能です。現代においては指数関数的に増加する「相互接続性の増大」という複雑さは企業の IT リスクとして扱われています。

対外システムへの接続テストがいかに困難であるかを以下に示します（**図 1**）。

例えば、対外システム数が「3」の場合、それらがお互いにつながるために必要な接続数は「3」になります。また、対外システム数が「5」の場合は、その必要接続数は「10」になります。これらの対外システム数と接続数の関係は以下の式で表現できます。

$$\text{接続数} = \frac{n \times (n-1)}{2}$$

※n：対外システム数

この数式から、接続システムが 1 つ増加するだけで、システム全体として検証が必要な接続数が飛躍的に増加することが分かります。これらを踏まえて、どのようなテストが必要かということが重要になってきます。

対外システム接続のテストにおいては、接続先のシステムが存在しない場合の「コンカレント（並行）開発」など、さらに状況が複雑な場合もあります。これは、複数プ

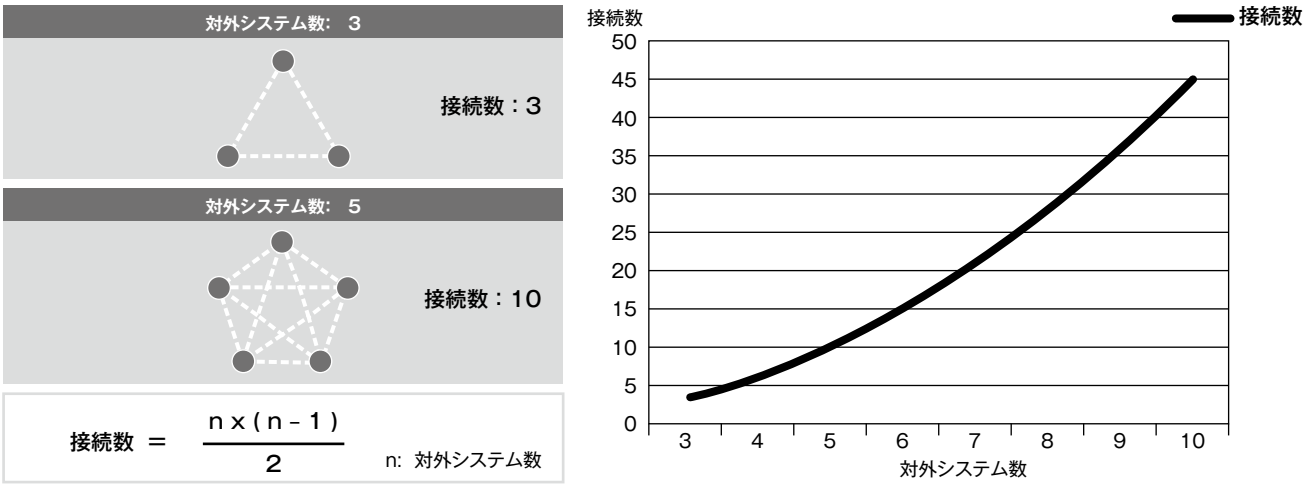

図 1. 対外システム数と接続数の関係

ロジェクトまたは複数チームが同時に開発を進行し、相互に一定の機能が完成した時点で合流するスタイルの開発です。

このスタイルでは、「相手がいない」接続をテストする必要性がしばしば発生し、特別な策を講じない限り、前述の「待ち」が発生するため、並行開発を進めても効果的な開発スケジュールの短縮につながりません。さらには一斉テスト時の初歩的なミスによる作業遅延を予防するためにも事前検証をできるだけ早い段階で行う必要性が出てくるでしょう。

2.3 性能テストや負荷テストをいつも最後に実施する

一般的には、アプリケーション開発工程の終盤において性能テストや負荷テストが行われていますが、最終的な確認のつもりで行ったテストにて問題が発覚した場合、リリースに向けて計画通りに作業を進めることが難しい状態に陥ります。問題点の改修にかかる時間やコストをどのように調整すべきか追加のプロジェクト計画が必要となり、さらには大幅な修正が困難と判断した場合は、割り切り事項を合意することで解決とすることもしばしば見受けられます。

昨今のシステム信頼性や性能要件の高まりに比して、複雑度も増加していることから、終盤でのテストでこうした障害が発生するケースが増えています。これら障害は単にテスト不足に起因する場合に多く、機能性充足のためのテストに偏重している場合に頻発します。

性能低下や突発的な負荷に対応できないといった低信頼性につながる課題の多くは、複雑性よりも不規則性（例: ボトルネックがどこに発生するか“読めない”状況に陥るなど）に原因があるといえます。

❸ テスト仮想化とは?

前述の通り、「結合を部分的・段階的に行うテスト」の必要性、「相手がいない接続テスト」の必要性、および「全部結合する以前、トランザクションが疎通する以前の性能・負荷テスト」の必要性がテストに関する期待と要件を高度化させています。このような背景から生まれた考え方が「テスト仮想化」になります。

テスト仮想化は、昨今推進されているシステム資源の仮想化とは異なり、結合・接続テストにおける相手先の「仮想化」を意味します。

テスト仮想化は、テストの期間を短縮し、実施コストを低減し、リスクを低減する方法として、今後ますます注目される技術です。

具体的には以下のような技術によって実現されます。

1）シミュレーション用の「ドライバー」および「スタブ」を用いて、テストを実施可能な単位に分割する技術
2）シミュレーション・スタブを用いて、存在しない相手先への接続テストを行う技術
3）ドライバーとスタブで挟んだ「部分結合対象」について性能測定や負荷テストを可能にする技術

一般的にはドライバーやスタブは、その製造コストの高さからすべてのコンポーネントに対して作成することはありません。また、従来の静的なドライバーやスタブでは、仕様変更時に保守の対象となり、保守運用対象が増加

するため、可能な限り自前で作成したくないものです。さらに、ドライバーやスタブの利用は結合テストの「回数」をいたずらに増やし工数が増加する感覚があるため敬遠されがちですが、実際には分割した結合単位ごとのテスト成功確率が高まります。待ち時間の短縮および手戻りの最小化と残存欠陥の検出漏れを防止できる観点から、ほとんどの場合ビッグバン・テストよりも工数と期間が短縮されます。

よって、テスト仮想化の実現には変更容易性と保守性の高いシミュレーション・スタブやドライバーを安価に生成する技術が不可欠であるといえます。

シミュレーション・スタブやドライバーの適用は、分割統治や構造化といった複雑度に対処する設計原則の適用にほかならず、人間が計測・管理可能な単位へ切り出してテストする技術になります。

しかしながら、昨今の短納期開発の要求からは、シミュレーション・スタブやドライバー作成に投入する工数も期間も捻出できず、相手先の完成を待って一斉に検証を行うことが主流です。このようにビッグバン・テストが日常的に行われているため、開発者およびテスト担当者は「それしか方法がない」錯覚に陥っていると想定されます。

❹ テスト仮想化の将来

では、テスト仮想化はどのように活用されるべきでしょうか。以下に例を挙げて考察していきます。

1）旧バージョンの保存による環境再現コストの低減

派生開発や度重なるシステム改変においては、「過去のバージョンから存在する欠陥か?」「旧環境から潜在していた欠陥か? 」など、過去の環境を再現してテストする場面も少なくありません。旧システムから移行したシステム環境や、過去に使っていたパッケージをシュミレーションしたスタブがあれば、仮想的にテスト環境が再現できます。その環境が極めて短時間に構築できれば、システムを初期開発時からサンセット（引退）させるまでのすべての改変を保存し利用することも可能となるでしょう。

2）テスト・プロセスの変化

ユニット・テストでは終わったものから順次結合していくことが可能になります。それに伴い、結合したものから順次性能測定や負荷テストを実施できるなど、各テスト・レベルが同時並行で実施することが可能となると予想されます。これは開発プロセスに変化をもたらし、従来の「フェーズド・アプローチ」の考え方に変化を与えるものになります。

この場合の課題は、高度なテスト管理プロセスの導入が必要になることと開発者に対してテストや品質に対する高い知識や経験が要求されることです。

3）ソースコード実装前に、シミュレーション・スタブを提供

マルチベンダー環境においては、ソースコード開発前にシュミレーション・スタブを提供することで、他ベンダー間の依存性を低減することができます。これには 2 つの利点があります。1 つ目は、他ベンダーの進捗や変更リスクが自システムに波及しないようにして、依存性を低減できること。2 つ目は、自システムの制約や仕様変更をほかの個所に影響させないことです。

これもインターフェース確立と構造化設計による複雑性低減策の 1 つになります。

4）オフショア開発における「仕様充足」のためにスタブを提供

国をまたがったオフショア開発においては、本番環境への直接接続や自社以外の対外接続の前に安全性・信頼性を確認したい場合が少なくありません。

またリモート環境において発注した機能仕様を充足させることは、ユニット・テスト・レベルでは比較的簡単ですが、既存のコンポーネントの接続やサブシステム間インターフェースの接続テストにおいては、容易には実施できない場合があります。

このような場面においてもスタブ・ドライバーを安価に生成して配布する必要性が発生してくるでしょう。

❺ IBMが提唱する今後のテスト方針

テスト仮想化がもたらす価値の重要性やその必要性について、これまで述べてきました。IBM としてはテスト仮想化の普及に先駆けて、2012 年 6 月に仮想的なテスト環境を自動構築する「IBM Rational テスト仮想化／自動化ソリューション」[3] を発表しました（**図 2**）。

本ソリューションでは、アプリケーションの品質保持と同時にテスト期間の短縮およびコスト削減を実現します。企業内外にかかわらず、システム間連携は増加するばかりであり、さらに基幹システムやクラウド、モバイル端末など連携するシステムの複雑度も同様に増しています。このような相互依存関係にある昨今のアプリケーションでは、

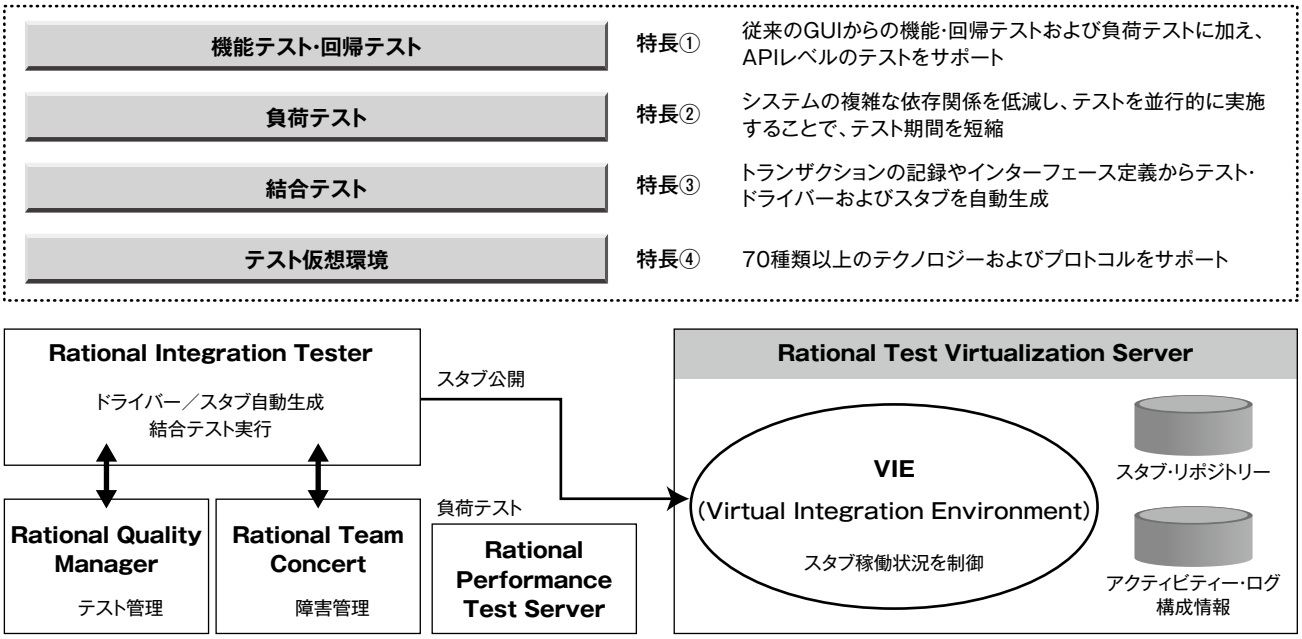

図 2. IBM Rational テスト仮想化／自動化ソリューション

個々のプログラムの品質が接続するアプリケーションおよびシステム全体に大きな影響を与えてしまうことは言うまでもなく、いかにして膨大な組み合わせの結合テストを効率よく、短期間で完了させるかが重要な鍵になってきます。

1）品質管理ソリューションの強化

IBM ではユーザー・インターフェース（以下、UI）に対するテスト自動化ソリューションにて、機能テスト・回帰テスト・ツールとして「Rational Functional Tester（以下、RFT）」を、負荷テスト・ツールとして「Rational Performance Tester（以下、RPT）」を提供しています。

今回の発表では上記テスト自動化ソリューションに API レベルの結合テスト（インターフェース・テスト）をサポートするソリューションを加えました。

IBM Rational Test Workbench：

複雑化したアプリケーションの品質に関する課題に対応するため、機能テスト・回帰テスト（RFT）、負荷テスト（RPT）、および結合テスト「Rational Integration Tester」を提供します［4］。

IBM Rational Performance Test Server：

アプリケーション・レベルおよび統合レベルの負荷生成機能を組み込み、すべてのコンポーネントにわたるパフォーマンスと拡張性を完全に把握します。

IBM Rational Test Virtualization Server：

システムの振る舞いのモデル化およびシミュレーションを行い、アプリケーション・テストの依存度を排除します。また、インフラストラクチャー・コストの削減を実現します。

結合テストでは UI の完成を待つ必要がなく、従来よりも早くテストを開始でき、UI テスト時に不具合が発覚した場合に比べ修正コストを抑えつつ、スケジュール遅延を防止します。これまでも大きな効果を実現してきた打鍵の自動化やテスト・バリエーションのテスト自動化は主にシステム・テストや保守フェーズにて利用されることが多いのに対して、新ソリューションはユニット・テストや結合テストの段階から開始することが可能となります。これにより、アプリケーション開発におけるテスト工程をさらに幅広くサポートできるソリューションとなりました。

2）アプリケーション・ライフサイクル・マネジメントとの親和性

ソフトウェア開発におけるアプリケーション・ライフサイクル・マネジメントは、ビジネス要求とその目的に照準を合わせ、各種成果物間のトレーサビリティー確保など要求管理から開発管理およびテスト管理までを一元的に実現する環境を提供するものであり、IBM ではそのような開発環境をコラボレーティブ・ライフサイクル・マネジメン

ト・ソリューション（以下、CLM ソリューション）として定義しています。今回ご紹介した「IBM Rational テスト仮想化／自動化ソリューション」は、CLM ソリューションの構成要素であるタスク管理や障害管理を担う IBM Rational Team Concert およびテスト・品質管理をつかさどる IBM Rational Quality Manager と密に連携し、ソフトウェア開発プロセスとの親和性を保ちつつ、さらに強化するものとなっています。

3）適用効果

本ソリューションは、すでに多くのお客様に採用されています。大規模システムになればなるほどすべての環境をそろえてテストできる期間は限られており、それらの準備に掛かる人的リソースや期間の範囲内にテストが完了せずリリース遅延につながるリスクなどを考慮すると、その 1 回のテストに掛かる工数は計り知れないものとなります。このようなリスクを最小限に抑え、生産性の高いテストを実施するためにもテスト仮想化が必要となってきます。

ある金融機関のお客様では、複雑なシステム環境に対して本ソリューションを適用し、プラットフォームをまたがったテストを実現することで、アプリケーション検証に必要な時間を大幅に削減することができました。また、ある通信業界のお客様では、通常 4 週間ごとにテスト・サイクルを回していましたが、そのサイクルを 1 週間未満にすることができ、従来に比べ 3 倍以上の生産性向上を実現しました。さらに、テストが完了するまでのタイム・フレームは、計画時に見積もりした時間から最大 75% 削減することができました。

❻ まとめ

今回はソフトウェア・テストにおける生産性を阻害する要因とその解決策としてのテスト仮想化を紹介しました。リーマン法則の第 2 法則の通り、特別な施策を講じない限り、われわれが日々利用するシステムは今後も複雑化するばかりです。よって、その複雑さとどのように向き合い、どのようにうまく付き合うかが重要になってきます。IBM では、日々変わりゆく IT 環境の変化や今後の方向性を踏まえ、個々のお客様が抱える課題に対して、その時々における最適なソリューションを提供できるよう取り組んでいます。

［参考文献］

［1］ Lehman, M. M: "On Understanding Laws, Evolution, and Conservation in the Large-Program Life Cycle," Journal of Systems and Software 1: 213?221. 10.1016/0164-1212(79)90022-0 (1980).

［2］ IBM Global CEO Study 2010 “Capitalizing on complexity,” http://www.ibm.com/services/us/ceo/ceostudy2010/index.html (2010-05).

［3］ 日本アイ・ビー・エム株式会社 : “プレスリリース : システム開発のテスト期間とコストを削減,” http://www.ibm.com/jp/press/2012/06/2101.html（2012-06）.

［4］ Rational テスト仮想化 / 自動化ソリューション - Test Workbench, http://www.ibm.com/software/jp/rational/products/rtw/

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
ワールドワイド・タイガーチーム

細川 宣啓 **Nobuhiro Hosokawa**

［プロフィール］

1992 年日本 IBM の SI 部門に入社。各種のアプリケーション開発プロジェクトを経験した後、同社品質保証部（QA 組織）にて品質レビュー組織「Quality Inspection」を設立。約 10 年の品質検証経験を生かし、現在は Rational のワールドワイド・タイガーチームに所属。品質系技術分野をリードする。
IEEE Associate メンバー、日本科学技術連盟 SQiP 研究会主査、NPO 法人 ASTER Project Fabre（プロジェクト・ファーブル）代表。早稲田大学 非常勤講師。

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ズ
主任 IT アーキテクト

金元 隆志 **Takashi Kanamoto**

［プロフィール］

1999 年日本 IBM 入社。サービス部門にて大手都市銀行のプロジェクトに SE として参画。専門はアーキテクチャー設計技術。2004 年から先進技術検討および適用に向け、海外 IBM と協業。2009 年よりソフトウェア事業ラショナル事業部に異動。ソフトウェア・ライフサイクル全般のアプリケーション開発支援に従事。ISO/IEC JTC1 SC7 WG4 委員、NPO 法人 ASTER テスト Tool WG メンバー。