ソリューション技術通信　ベリサーブナビゲーションでは、さまざまな分野で検証に携わるベリサーブが「今とこれから」を皆様にお届けします。

VERISERVE NAVIGATION

VOL. 13
2018 March

特別寄稿

ソフトウェアにおけるUIデザインの重要性

植松 清 氏

特 集

ソフトウェアテストのニューノーマル
～テストの新たな常態・常識～

辰巳 敬三 氏

連載（第3回）

情報システムの脆弱性に関する法律問題

松島 淳也 氏

イベント参加レポート

現新比較検証サービスのご紹介

VERISERVE

ソフトウェアにおける UIデザインの重要性

特 別 寄 稿

植松　清氏（うえまつ きよし）

産業能率大学および桑沢デザイン研究所卒業
インダストリアルデザイナーやマルチメディアクリエーターを経て
1993年フェノメナエンターテインメント株式会社設立。

この5年間でUI（ユーザーインターフェース）デザインに対する関心が日本のIT業界で急速に高まっています。

コンシューマサービスだけでなく、業務用システムや産業機器についてもUIデザインに配慮した操作画面の開発が求められる時代になってきました。

しかしながら、何をどうすれば優れたUIデザインを作り上げることができるかについては、まだまだ暗中模索が続いているのが現状です。

本稿では、UIデザインの現場に従事しているデザイナーの視点で「技術者が取り組むべきUIデザイン」について述べさせていただきます。

UIデザインに対するニーズの高まり

私たちの会社では、数年前から産業機器や製造設備開発企業からのUIデザインに関するお問合せが増えてきました。

お問合せをいただいた際、何故UIデザインを取り入れる必要が発生したのかを必ず質問するようにしています。お答えは総じて以下に大別することができます。

- 営業部門から「自社製品の操作画面が他社に比べて見劣りがする。なんとかして欲しい。」という要求がある。
- 会社の上層部から「UX・UIデザインというものを盛り込むように」と指示を受けた。
- お客さまからの入札要件に「高い操作性の実現」という項目が存在している。

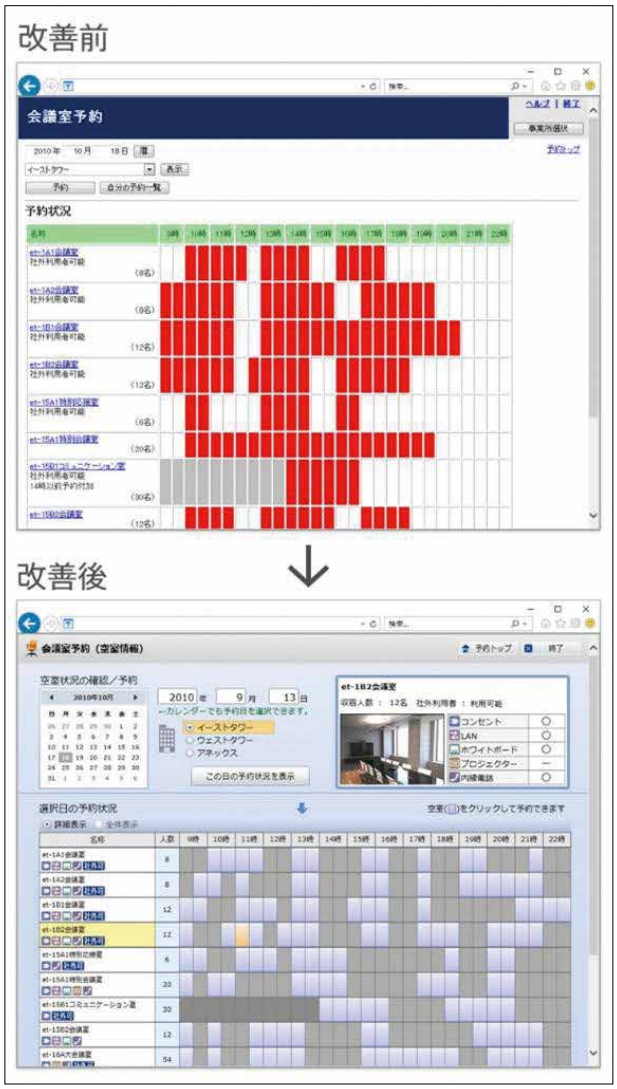


図1　UIデザインの比較調査サンプル

残念ながら多くの場合、開発現場による自発的なUIデザインへの取り組みではなく、外因によるものです。

開発現場が気づかないうちに、エンドユーザー側のUIデザインに対するニーズが高まってきています。

エンドユーザー側がUIデザインを求めて始めている利用は様々だと思いますが、労働現場の「高齢化・外国人化・短期就労者化」に対応するため、機器の操作性を向上させたいというニーズを頻繁に耳にします。

IoTやAIを活用した生産性向上を実現するのにはまだ時間と投資が見合わないため、既存設備の改善や少ないIT投資で対処したいというものです。

確かにUIデザインの品質向上は生産性向上につなげることができます。

両者の関係を立証するために、会議室予約システムに関するUIデザイン改善の効果測定を実施したことがあります。改善前は技術者が社内システム向けに構築していた「わかりにくく、操作に手間のかかる」操作画面でした。そのシステムに対してサーバー側の処理系は変更せず、画面の見た目と操作手順だけを見直したリニューアルを行いました。（図1）

見た目については、彩度を抑えた色使いに変更したことで文字部分に視線を集中しやすくするなどの改善を行いました。

操作手順の改善として、各会議室の設備情報等を第2階層画面にテキストで表示していたのを、アイコン化して第1階層画面上に表示するなどの工夫を行っています。

ユーザーの操作時間を新旧で測定すると、改善により約50%に短縮されました。（図2）

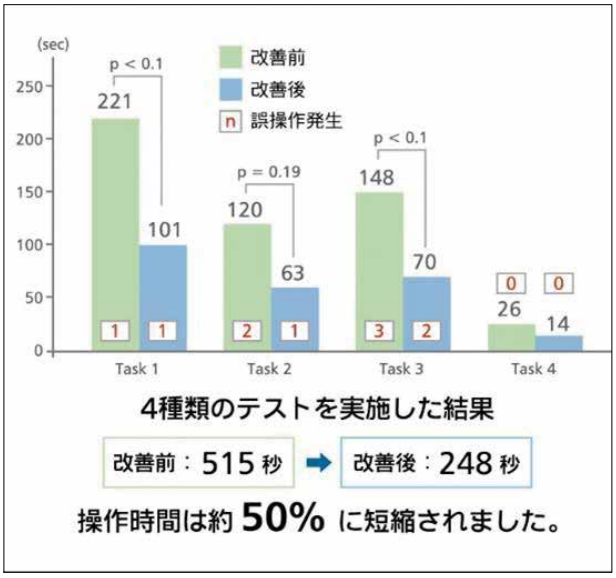


図2　UIデザインの比較調査結果

100回の会議室予約操作を行なった場合は約1.6時間の労働時間短縮になります。大企業全体の労働コストに換算するとかなり大きな削減になります。

業務用システムのUIデザインについては「ユーザーによる操作性」に配慮を行っていないものがほとんどです。「働き方改革」で労働生産性の向上に対する関心が高まっている中、ITシステムのUIデザイン改善が生産性向上に繋がることに気付いた企業も増えてきています。

今後は、eコマースのようなコンシューマ向けシステムだけでなく、業務用ソリューション分野でもUIデザインに配慮しなくてはならないケースが増加していくと考えられます。

UIデザインは複雑

実際にUIデザイン業務に取り組むと、その複雑さに戸惑う方が多いです。

プロセスは数段階で済みますが、各プロセスを遂行するために必要とされる知識と経験が多岐に渡っています。(図3)

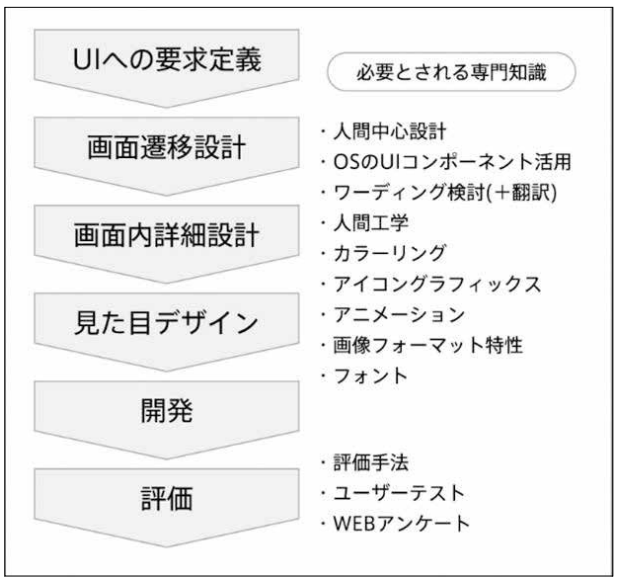


図3　UIデザインの工程と専門知識

UIへの要求課題を把握するためにはユーザーの声を集めて分析する能力も求められます。設計時には人間工学に関する知識も必要です。美しくてモダンな画面表示を実現するためにグラフィックデザイナーと共同作業する必要もあります。当然ですがOSやデバイス、ソフトウェア開発環境に対する知識も求められます。

以前は「アート系の学生バイトにお願いして画面のデザインをしてもらう」みたいな話がよくありましたが、現在そのレベルではユーザーの求める品質を実現できなくなっています。

Apple, Microsoft、Google、Amazonといった米国のIT企業ではUIデザインを事業推進の重要な要素と捉え、様々な分野の優秀な人材を大量に投入しています。

UIデザインは時間をかけて経験と知識を習得していく必要があります。具体的な開発案件が決まってから手をつけても間に合いません。技術者はUIデザインを新しい基礎技術の習得と捉えて、自発的な取り組みを開始していくことが必要です。

ユーザビリティの評価

UIデザインに対する関心の高まりとともに、UIデザインの品質を評価したいというニーズも当然増えています。

しかしながら、ほとんどの場合「使いやすさの指標」が存在していないため、評価を実施しようとすると四苦八苦します。

そもそも「使いやすくて優れたUIデザイン」というものに対する明確な指標は存在していません。UIデザインについてはインターネットや書籍でいろいろなノウハウが紹介されていますが、どれもTIPS的な内容が多くて意外と役に立ちません。

「これとこれを、こうしておけば普遍的に使いやすいUIデザインになる」というシンプルな指針は存在しないようです。

UIデザインに関係した指針化に取組むと「ニールセン」という言葉を必ず耳にします。「ニールセン」とはユーザビリティエンジニアリング研究の第一人者であるヤコブ・ニールセン氏のことです。ニールセン氏は「使いやすい」という曖昧な事象を「ユーザビリティエンジニアリング」という考え方で科学的に研究し、実務に活用できる資料を大量に公開しています。

「使いやすくて優れたUIデザイン」を作り上げる、評価する際にはニールセン氏の提唱する「ユーザビリティエンジニアリング」がとても役に立ちます。

以下はUIデザイン従事者の間で「ニールセンの10項目」などと呼ばれ尊重されています。

- シンプルな自然の対話を提供する。
- ユーザーの言葉を使う。

・ユーザーの記憶負荷を最小限に留める。
・一貫性を保つ。
・フィードバックを提供する。
・出口を明らかにする。
・ショートカットを提供する。
・適切なエラーメッセージを行う。
・エラーを防ぐ。
・ヘルプとドキュメンテーションを提供する。

各項目の意味するところは、東京電機大学出版局「ユーザビリティエンジニアリング原論」（ヤコブ・ニールセン著）をお読み下さい。

上記書籍には「使いやすさ」の評価手法についても詳しく解説されており、UIデザイン業務に従事する全ての人への教科書としてとても有効です。

見た目の「かっこよさ」を評価する

UIデザインという言葉には「使いやすさ実現する」というエンジニアリング寄りの目的だけでなく、「魅力的で美しい見た目にする」というアート寄りの目的が求められるケースが多く存在します。

10年位前までは「UIデザイン業務 ＝ 斬新で美しい見た目を実現する業務」と捉えている人がほとんどでした。現在では見た目のみを重視したUIデザインは減りつつありますが、「見た目」もUIデザインにおける評価対象として重要です。

「見た目の善し悪しは判断できない」と言う技術系の人がほとんどです。確かにセンスを求められる「芸術性」についての評価は困難だと思いますが、「高品質な表示物による美しさ」は技術者でも評価可能です。

例えば「整理整頓」です。美しい見た目の画面は整理整頓されています。整理整頓された見た目は安心感や信頼感をユーザーに与えることができます。（図4）

「見た目」についても感性に頼らずに品質評価できる項目は存在しています。「デザイン」という言葉をブラックボックス化せず、評価範囲を明確にしていくことが必要といえます。

UIデザインの今後

スマートフォンが普及し、AppleやGoogleがUIデザインに力を注いでいる現在、ユーザーがソフトウェアに求める操作性の品質基準は高まるばかりです。

処理機能や性能を満たしていれば操作性は低くても構わないという考え方が通用しなくなってきています。

音声応答やAI、VRに対応するUIデザインも必要になってきました。デザインに対する品質評価の実施も増々重要になります。

今までは「UIデザインはデザイナーさんの担当」で「デザイナーさんはなんとなく芸術っぽい人」という考えが強かったのですが、今後は技術系の人がUIデザイン業務の主体になるケースが増えていきます。

UIデザインはいまだに開拓途上の工学分野です。ITに関わっている方はスキルアップの一つとして取組んでみることをお勧めします。

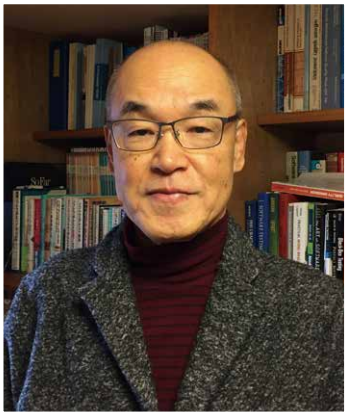
図4　見た目の整理整頓

ソフトウェアテストのニューノーマル
～テストの新たな常態・常識～

特　集

辰巳　敬三 氏（たつみ けいぞう）

【経歴】
1976年、富士通株式会社に入社。ソフトウェア製品検査部門でメインフレームOSの検査、品質保証を担当。その後、UNIXやPCのソフトウェア検査に従事。
1999年に検査の現場は離れたがソフトウェアの品質、
テストの技術の探求をライフワークとして研鑽を続けている。
2009年から2016年にNPO法人高度情報通信人材育成支援センター（CeFIL）に出向し産学連携のIT人材育成を担当。
【主な著書・講演】
・富士通のソフトウェア品質保証活動（共著）
・ソフトウェア品質知識体系ガイド -SQuBOK Guide-（共著）
・初級ソフトウェア品質技術者資格試験（JCSQE）問題と解説 第2版（共著）
・テスト自動化クロニクル（JaSST'16 Tokai 基調講演）



〈はじめに〉

ニューノーマル（New Normal）は、以前なら考えられなかったようなことが当たり前の状態になることを意味する言葉です。この言葉が使われるようになったきっかけは、2008年のリーマンショックだそうです。当時の米国ではあらゆる経済指標が少し前には考えられなかった異常な水準まで落ち込み、こうした状態が一時的なものではなくこれからも長く続く「新しい普通の状態」という意味で、ニューノーマルという言葉が盛んに使われました。（「朝日新聞グローブ 2017年12月3日号 豊かさのニューノーマル」より）

ソフトウェアの世界でもアジャイル開発、DevOps、継続的インテグレーション/デリバリー (CI/CD)など、以前なら考えられなかったようなことが当たり前の状態（常態）になってきたのではないでしょうか。

本稿では、ソフトウェアテストにおいてニューノーマルになってきたアプローチや技術を紹介するとともに、その背景について考えます。

1 ニューノーマルへの流れ

本題に入る前に2000年頃からのエポックメイキングな製品やサービス、それらを支える開発技術やツールの出現状況を図1の年表で確認し、テストの常識が変わってきた背景を考えてみましょう。

1990年代前半に開発されたインターネット技術は、90年代後半にはAmazon、Yahoo、eBay、Googleなどの企業を産み出しました。2000年前後からはSalesforceのようにインターネットでビジネスアプリケーションのサービスを提供するビジネスが本格化し、クラウド・コンピューティングの時代になっていきました。

2000年代後半にはiPhoneやAndroidが出現、Mobileによるインターネット利用が広がり、同時期にサービスが拡大し始めたFacebookやtwitterなどのSNSとあいまって利用者個々人が容易につながるようになりました。企業側から見ると顧客接点の手段が飛躍的に高度化できるようになりました。これらのデジタルテクノロジーは総称してSMAC（Social, Mobile, Analytics, Cloud）と呼ばれています。

そして、2010年前後にはAirbnb（宿泊施設予約サービス）やUber（自動車配車サービス）のようなデジタルテクノロジーを活用して既存ビジネスを破壊する新興企業が出現し拡大し始めました。デジタルテクノロジーを活用した革新的なビジネスモデルはデジタルビジネスと呼ばれ、従来企業においても自らの既存ビジネスをデジタル化して革新するデジタルトランスフォーメーションの動きが活発になっています。例えば、IBM社が2015年に実施した世界の経営層へのインタビュー調査結果では「世界のCEOは、テクノロジーによる業界の再定義とビジネスのデジタル化に強い関心を示している」とまとめています［1］［2］。

ソフトウェアやシステム開発技術の観点から見てみます。2000年前半にアジャイル手法が広まり始め、TDD（テスト駆動開発）やBDD（ビヘイビア駆動開発）などの技術やスクラムなどのアジャイルソフトウェア開発方法論の普及により近年では多くのプロジェクトでアジャイル手法が採用されるようになりました。

また、システム開発や運用の基盤として仮想化技術をベースにクラウドの仕組みが実用化され、2006年には当時のGoogleのCEOのEric Schmidt氏　が「Cloud Computing」と呼ぶに至りました。同年にはAmazonが自社のクラウド基盤のノウハウをもとにAmazon Web Servicesの提供を開始

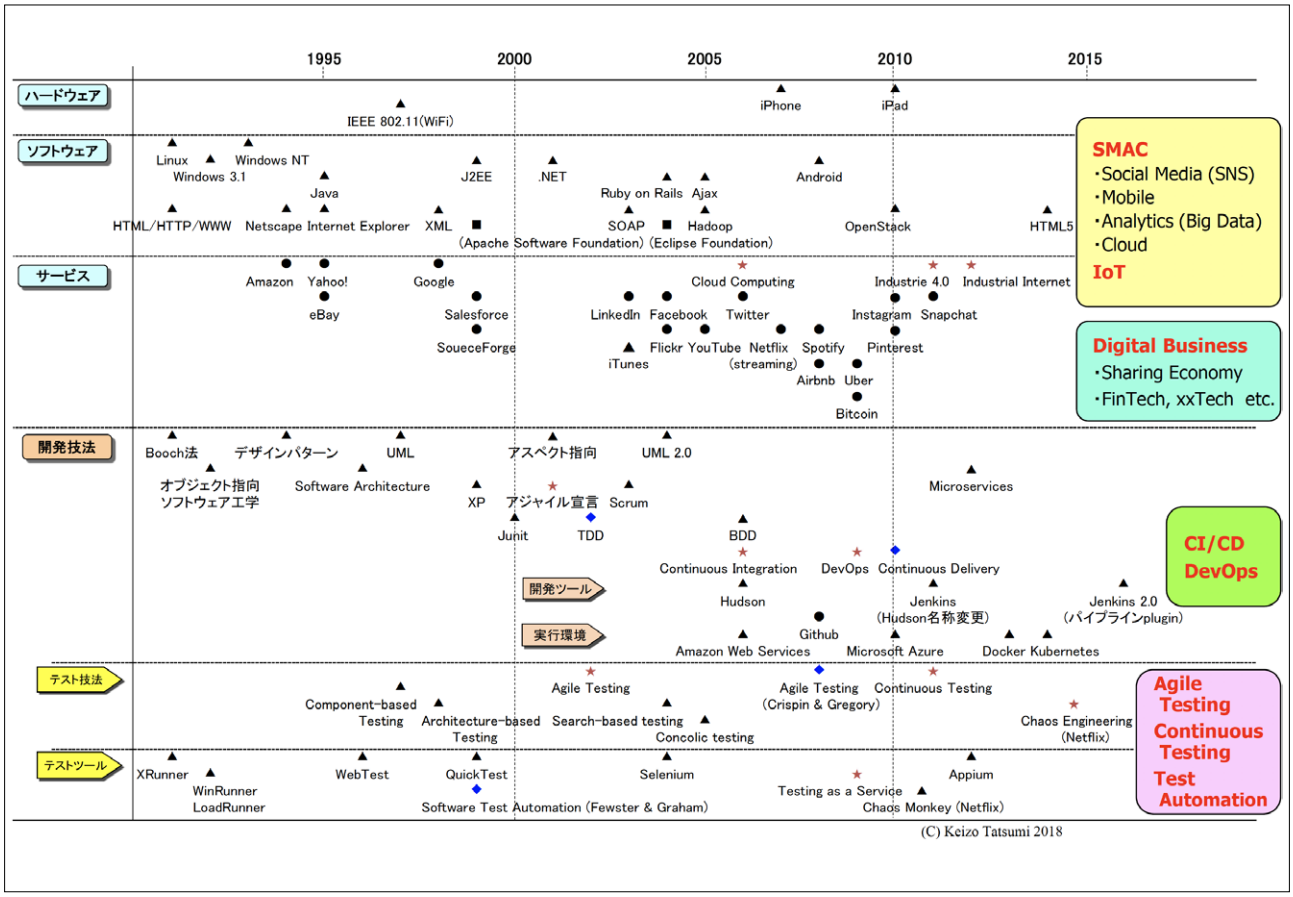


図1. ソフトウェアテスト年表（ニューノーマルへの流れ）

し、現在は世界のクラウド市場のトップとなっています。2010年代に入ると仮想的なアプリケーション実行環境を構築するコンテナ技術が注目され、2013年に登場したDockerは効率の良い環境が構築できることから普及し始めています。

アプリケーション開発では、コード変更のたびにコンパイル、テスト、静的解析という一連のビルド作業を行ってソフトウェアを常に統合する継続的インテグレーション、その後の本番ソフトウェアの作成や本番システムへの展開（デプロイ）までも行う継続的デリバリーや継続的デプロイの手法が、これらの作業を支援するJenkinsなどのツールの登場により普及が拡大しています。

以上のように図1から、デジタルビジネス時代を生き抜くための素早いビジネス展開の要請に呼応して、開発側の技術がめざましく進展していることが見て取れるのではないでしょうか。さらに近年は運用側と連携してシステム開発を進めるDevOpsが提唱され、顧客の望むシステムをより迅速に提供・改修できる仕組みの実現、実践が始まっています。

このような状況の中で品質保証やテストへの取り組み方も変わってきました。コードが出来上がってからテストを開始するのではなく、開発の始めからデリバリーまでに行われる協働的なテストを実践するAgile Testing［3］、開発の上流からテストを継続的に行う継続的テスト、継続的な活動を支えるテスト自動化は変化してきた取り組みの例です。Capgemini社が毎年実施している世界の企業の品質やテストへの取り組み調査World Quality Report 2017年版では品質保証とテストの最新動向が以下の項目に沿ってまとめられています［4］。

- デジタルトランスフォーメーション
- アジャイルとDevOps
- テスト自動化
- 工業化
- テストデータとテスト環境のマネジメント
- 品質保証とテストの予算

これらに対応していくために、以前なら考えられなかった取り組み方が当たり前に実践される時代になってきたということでしょう。

2 テストのニューノーマルの特徴

ここでテストの分野でニューノーマルになってきたアプローチや技術にどのような特徴があるのかを考えてみましょう。

Techwell社[1]シニアコンサルタントのMichael Sowers氏は、アジャイルで継続的にインテグレーションやデリバリーが行われるデジタル時代におけるソフトウェア開発やテストのニューノーマルの特徴として次の10項目（筆者訳）をあげています［5］。

(1) 開発とテストはチームスポーツ
(2) データとアナリティクスの役割の増加
(3) テストと開発の協調（TestDev）の考え方の拡大
(4) 全て継続すること（Continuous everything）は全て自動化すること
(5) 稼働中テスト（Testing in production）は珍しいことではない
(6) より深いスキルセットが必須
(7) 自動化の拡大
(8) チーム全体の責任
(9) ほぼリアルタイムな測定やメトリクスが可能
(10) リスクの許容範囲の変化

ニューノーマルへの流れで紹介した開発技術の進展を見ると、これらの特徴付けはうなずけるのではないでしょうか。前述のテスト技法も、チームにおける協働的なテストを実践するAgile Testingは（1）（3）（8）、ライフサイクルに渡ったテスト活動を実践する継続的テストは（2）（4）（5）（9）、テスト自動化は（4）（7）の特徴をもつ取り組みと言えます。

[1] Techwell（旧社名はSQE）は品質/テストのトレーニングサービスの提供やSTAREAST/STARWESTなどのカンファレンス開催を行っている米国の企業

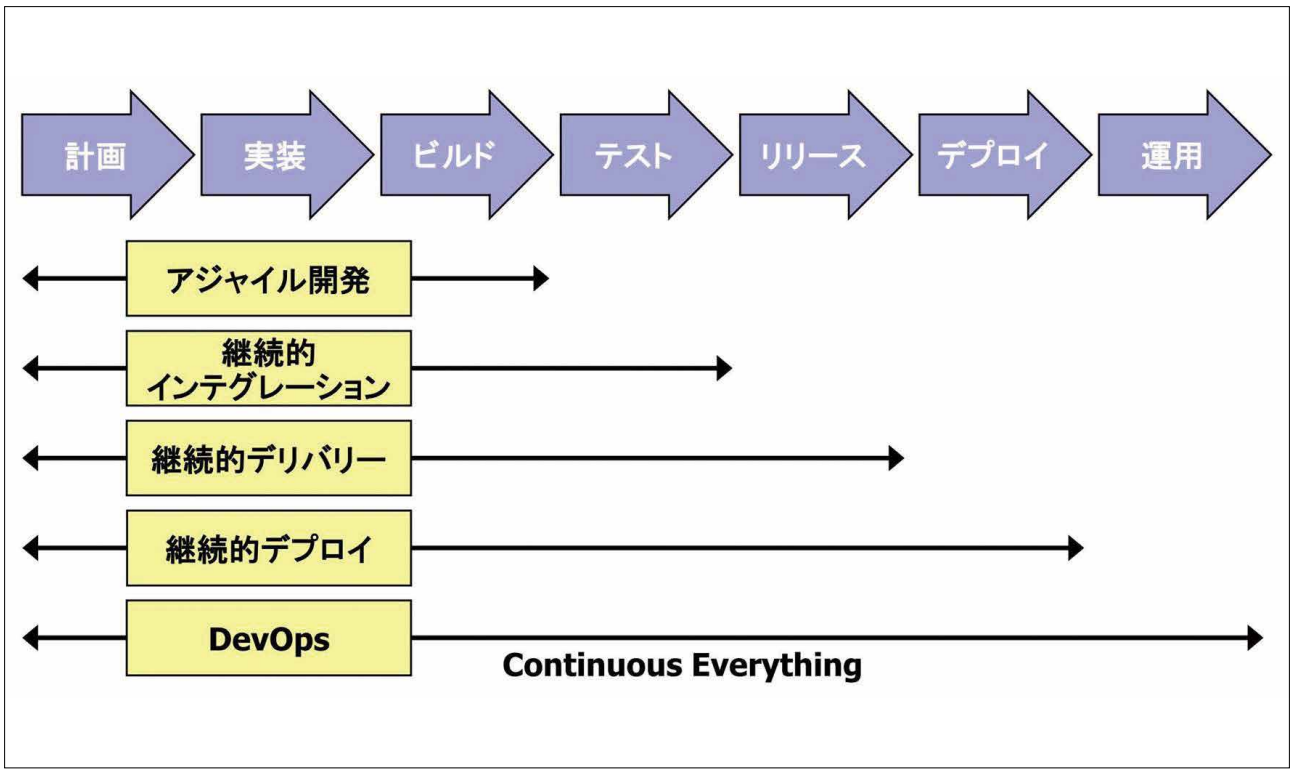


図2. DevOpsの要素技術の対象範囲

3 新たなテストのアプローチ

筆者はソフトウェアテスト技術の歴史に興味があり長年調査していますが、今ほど社会やビジネスシステムの動き、システムやソフトウェア開発技術と密接に連動して新たなテストのアプローチが生み出されている時代はないと感じています。経営側のデジタルビジネス化への意識の高まりやビジネススピードの要求を背景に、アジャイル開発の普及、さらには運用側まで含めたDevOpsの実践が始まり、ボトルネックになりがちだったテストの作業に焦点があたった結果だと思います。

ここで、DevOpsに至る要素技術の関係を見てみましょう。図2にDevOpsの要素技術が対象とする開発ライフサイクル

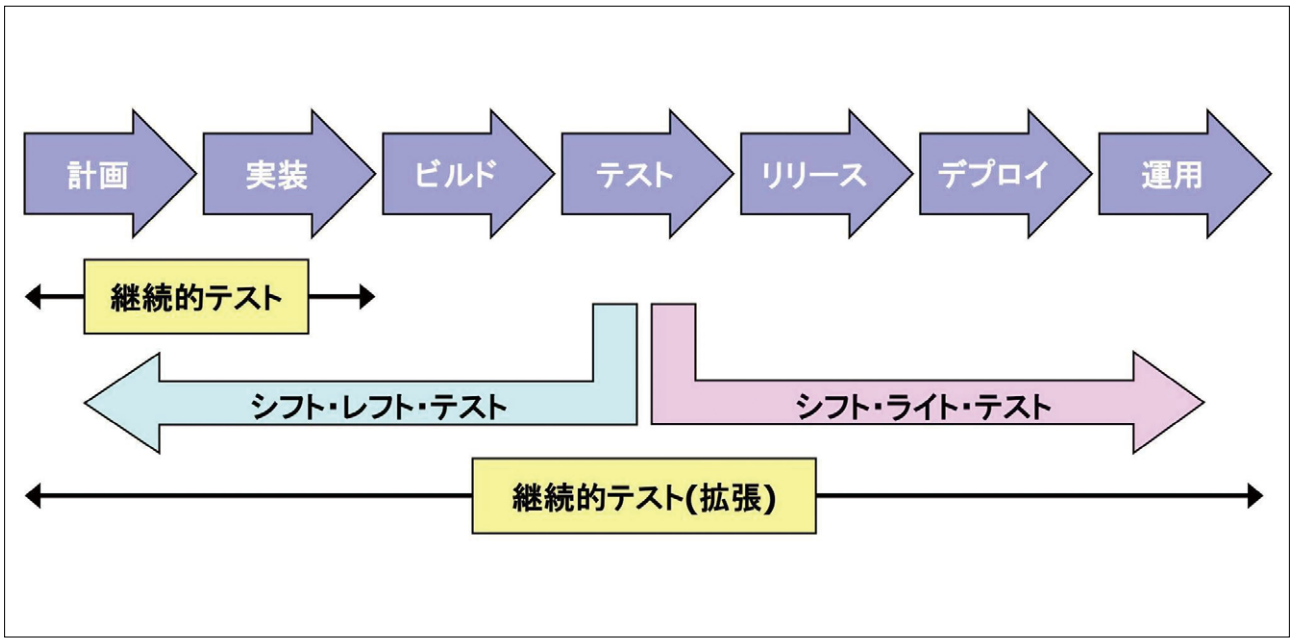


図3. テストのシフトと継続的テストの拡張

を示します。各技術の対象範囲を示す矢印の中はテストを含め自動化されることが前提です。例えば継続的デリバリーでは実装以降、ビルド、テスト、リリース版の作成が自動的に実行されます。継続的デプロイでは、さらに本番環境への展開まで含めて自動的に実行されます。そして、DevOpsになると運用まで含めて全てを継続的に行う（Continuous Everything）ことになります。

このように全てを継続的に、しかも早く行うことが必要な状況でテストにも継続的テストが求められるようになりました。そのため、開発の終盤で行っていたテストを開発プロセスの左側（上流）に移す「シフト・レフト・テスト（Shift Left Testing)」、リリースをもって完了していたテストを運用も含む開発ライフサイクルの右側（下流）に移す「シフト・ライト・テスト（Shift Right Testing)」のアプローチがとられるようになってきました（図3)。

これらの用語はDevOpsの拡がりとともに、欧米のカンファレンス、Blog、ITベンダーやテストサービス会社の記事で見ることが多くなりました。日本でも2014年頃から外資系ITベンダーの記事で継続的テスト、シフト・レフトを目にし始めましたが、シフト・ライトはまだほとんど目にしません。バズワード的な用語ではありますが欧米の記事を見る限り、支援ツールの充実により着実に実践が始まっているようです。

以降では、ニューノーマルになりつつあるテストの新たなアプローチとして継続的テスト、シフト・レフト・テスト、シフト・ライト・テストを紹介します。

3.1 継続的テスト

継続的テストという用語が使われ始めた当初は、開発の早い段階でテストに着手しテストを自動化して左側で繰り返しテストし続けるという意味合いだったようですが、現在は右側も含めた全ての局面でのテスト活動を指すように変わってきています（図3)。

例えばテスト自動化のリーディングベンダーであるオーストリアのTricentis社は、『継続的テストは、リリース候補版ソフトウェアのビジネスリスクに関するフィードバックを可能な限り迅速に得るために、ソフトウェアデリバリーパイプラインの一部として自動テストを実行するプロセスである。』という定義と14のキーポイントを示し［6]、キーポイントの一つとして以下をあげています。(『』内は筆者訳）『継続的テストはシフト・レフトからシフト・ライトまでのすべてを包含する。』この継続的テストの対象範囲の考え方は他社、例えばIBM社でも同様に定義されています［7]。

3.2 シフト・レフト・テスト

開発の上流からテスト活動を開始することの有効性は昔から言われており、例えば1980年には各開発工程に対応した活動が提案されています［8]。近年はWモデルという名称で開発工程と並行してテスト活動を行うモデルが紹介されています。

このようにテストを左側に移すという考え方自体は新しくありませんが、最近使われている「シフト・レフト」という用語は、開発がひととおり完了した後に行われるテストを、もっと早い開発の途中段階で、かつ自動的に行うことを意味していますので、Wモデルとは意味合いが異なっているところがあります。

シフト・レフトするためには、システム全体の動作を確認するエンドツーエンドのようなテストを開発コンポーネントが揃っていない状況で実行できるようにするという技術的課題があります。サービス仮想化技術（ツール）はこれを解決するものであり、コンポーネントの動作をシミュレートしてアプリケーション全体のテストが可能になるので、開発の早い段階で統合テストが実施できることになります。また、GUI経由のテストをAPI（Application Programming Interface）レベルで行うAPI testingによりテスト自動化の対象範囲が広まりました。

3.3 シフト・ライト・テスト

「シフト・ライト」はDevOpsが拡がる中で使われ始めた用語と思われ、概ね本運用（実稼働）に入ってからも本番環境でテストを行うことを意味しています。従来のテストのアプローチでは、リリース前に可能な限り多くのテストを実行し十分な品質であることの確信を得ることが基本的な考え方でした。しかし、システムの複雑化、利用者のニーズの多様化、あるいはパフォーマンスなど本番環境でしか確認できない事項の存在から、現実解として本番環境でのテストの必要性が高まってきました。Ops側（運用側）との連携の意識の高まりも要因としてあるでしょう。

シフト・ライト・テストの技術は総称して"Testing in Production（TiP）"と呼ばれることが多いようです。文字通り「本番環境テスト」です。学術系のテスト研究分野では"On-line testing"と呼ばれています。

TiPの主な手法に次のものがあります。

・カナリア・リリース（Canary release）

新旧2つのバージョンを同時に稼働させ、徐々に新バージョンを運用することで、新バージョンに問題が無いことを確認しながら移行するデプロイの手法です。手法の名称は「炭鉱のカナリア」に由来しています。

・A/Bテスト

デザインが異なる2つのWebページを用意し、ユーザーの利用状況（クリック率、導線など）の測定結果に基づいてWebページのデザインや導線の良さを評価し最適化を図る手法です。

・モニタリング

アプリケーションの性能、エンドユーザへの応答時間などの監視が元来の目的ですが、稼働中システムの動作結果を監視し入力ー出力／結果を解析するという点ではモニタリングとテストは同種のものと見なせます。テスト研究分野では、テストデータを用いる通常の能動的なテストであるActive Testingに対して、稼働中システムを監視して得た入力ー出力の解析結果に基づく受動的な評価という意味でPassive Testingと呼ばれています。

・Chaos Engineering

ネット動画配信のNetflix社では本番環境で意図的に障害を発生させ、システムの耐久性を確認するテストが日常的に行われています。この手法は障害注入テスト（FIT: Failure Injection Testing）と呼ばれ、Chaos Monkeyなどのテストツール群が開発されています。2015年にはこの手法を体系化したChaos Engineeringが発表されました［9］。

4 おわりに

テストの常識が変わってきている状況を紹介しましたが、読者のみなさんの現場ではいかがでしょうか。業種や業務によって変化のスピードは異なるかもしれませんが確実に変化の波はきているはずです。そして、今回紹介した「ニューノーマル」もいずれは「次のニューノーマル」に変わっていくでしょう。このように変化していく背景を理解し次の新たな常識を生み出していく力を育んでいきたいものです。

参考文献

[1] IBM グローバル経営層スタディ - 2015 境界線の再定義
https://www-935.ibm.com/services/jp-ja/studies/csuite/2015/

[2] IBMスタディ：世界のCEOは、テクノロジーによる業界の再定義とビジネスのデジタル化に強い関心を示している
http://www-03.ibm.com/press/jp/ja/pressrelease/48886.wss

[3] J. Gregory and L. Crispin, Our Definition of "Agile Testing"
https://agiletester.ca/definition-agile-testing/
【翻訳】「アジャイルテスト」、わたしたちの定義
http://www.kzsuzuki.com/entry/2017/07/06/234332

[4] World Quality Report 2017-2018, Capgemini/Micro Focus/Sogeti
https://www.sogeti.com/explore/reports/world-quality-report-2017-2018/

[5] Michael Sowers, The New Normal for Software Development and Testing, Better Software Magazine, Vol. 19, Issue 4, 2017 Fall

[6] What is Continuous Testing, Tricentis
https://www.tricentis.com/what-is-continuous-testing/

[7] M. Hollier and A. Wagner, Continuous Testing For Dummies, IBM Limited Edition, 2017
https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=KUM12367USEN

[8] M. A. Branstad, J. C. Cherniavsky, and W. P. Adrion, "Validation, Verification, and Testing for the Individual Programmer," Computer ,Vol.13, No.12, 1980

[9] クラウドのリージョンを丸ごと落とす過酷な試験を実現する「Chaos Kong」、Netflixが発表。「カオスエンジニアリング」の指針も表明, Publickey, 2015年9月28日
http://www.publickey1.jp/blog/15/chaos_kongnetflix.html

情報システムの脆弱性に関する法律問題

連載（第3回）

松島　淳也 氏（まつしま じゅんや）

【経歴】
1995 年 早稲田大学理工学部卒業　富士通株式会社入社
マイクロプロセッサーの開発、電子商取引システムの開発等に従事
2006 年 弁護士登録
2017 年 松島総合法律事務所設立

【主な取り扱い分野】
システム開発・ソフトウェア開発、システムの運用・保守をめぐる各種紛争処理
（請負代金等の報酬請求や損害賠償請求）及び契約事務
インターネットビジネスに関する各種紛争処理及び契約事務
知的財産権（特許権、著作権、営業秘密）に関する各種紛争処理及び契約事務

第1 はじめに

近年、情報システムの脆弱性を狙った事件･事故が多発し、個人情報の漏洩、ウェブサイトの改ざん等、情報システムを利用する企業に深刻な被害を及ぼしています。

情報システムの「脆弱性」とは、簡単に言うと不正アクセスや情報漏えいの原因となる情報システムの問題箇所のことです[1]。

前回（2018年1月号）取り上げたOSS（オープンソースソフトウェア）であれば、ソースコードが公開され、多数のエンジニアの目に晒されるため、脆弱性問題は発生しにくいと考える方もいますが、IPAへの届け出状況を確認すると、届け出された脆弱性情報は、約43パーセントがOSSに関する情報であることが公表されており[2]、OSSであれば脆弱性は問題にならないというわけではありません。

そこで今回は、①情報システムの脆弱性により発生した事例、②脆弱性が残る情報システムを提供したベンダの法律上の責任、③契約書等で規定されている損害賠償請求を免責・制限する条項の適用、④ベンダが脆弱性問題を回避するための工夫について解説します。

第2 情報システムの脆弱性により発生した事例

脆弱性に起因する事件・事故には、どのようなケースがあ

[1] 経済産業省告示第十九号（平成29年2月8日）では、「コンピュータウイルス、コンピュータ不正アクセス等の攻撃によりその機能や性能を損なう原因となり得る安全性上の問題箇所（ウェブアプリケーションにあっては、アクセス制御機能 により保護すべき情報等に不特定又は多数の者がアクセスできる状態を含む。）をいう。」と定義されています。

[2] IPA「ソフトウェア等の脆弱性関連情報に関する届出状況［2017年第1四半紀（1月~3月）］」の4頁参照。

るのでしょうか。ここでは、代表的なケースを3つご紹介します。

1.事例1（OpenSSLの脆弱性に関する事例）

最初にご紹介するのは、OpenSSLの「HeartBleed」と呼ばれる脆弱性に関する事例です。OpenSSLは、暗号化通信をするために広く利用されているOSSですが、「HeartBleed」[3]と呼ばれる脆弱性により、個人情報が漏洩する事故が発生しています。これは、暗号化通信をする際に、外部からの要求に対し必要以上のデータを送信してしまうという脆弱性であり、事例1ではこの脆弱性に起因して情報漏洩が発生したとされています[4]。

この事例の要点を整理すると、以下のとおりです[5]。

事例1の要点

①OpenSSL（暗号化通信に関するOSS）のHeartbleedの脆弱性を狙った攻撃によりクレジットカード会社の894人分の顧客情報が流出した。

②脆弱性の対策情報の公表日（4月7日）から不正アクセスを検知した日（4月11日）まで約4日という短期間で攻撃が行われた。

2.事例2（Apache Struts2の脆弱性に関する事例）

次にご紹介するのは、Apache Struts2の脆弱性に起因してウェブサイトが改ざんされてしまった事例です[6]。IPAによると、Apache Struts2は、ウェブアプリケーションを開発するためのソフトウェアフレームワークであり、日本国内で広く利用され、このフレームワークを利用して構築されたウェブサイトは相当数存在すると考えられています[7]。

この事例の要点を整理すると、以下のとおりです。

事例2の要点

①Apache Struts 2の脆弱性を突いた不正アクセスを受け、インターネット上で商品を販売する事業者のウェブサイトが改ざんされる。

②過去の注文者情報（氏名、住所等）も流出した可能性があるにも関わらず、攻撃内容がログに残らないようにされていたため詳細な原因究明ができない。

③最新のパッチは2日前に公開されていたが、パッチ適用には約1ヶ月間ウェブサイトを停止する必要があり、パッチ適用の検討中に攻撃を受ける。

④結果的に、ウェブサイトを約3ヶ月運用停止せざるを得ない状況になる。

3.事例3（SQLインジェクションの脆弱性に関する事例）

最後にご紹介するのは、WEB受注システムにおけるSQLインジェクションの脆弱性を攻撃されたため、顧客のクレジットカード情報が漏洩してしまったという事例です。この事例は訴訟に発展し、判決まで言い渡されたため、非常に注目を浴びた事案となりました。判決文の中で、SQLインジェクションとは「ウェブアプリケーションの入力画面にプログラム作成者の予想していない文字列を入力することにより、プログラム作成者の予想していないSQL文を実行させることである。」と認定され、更に、「このSQL文を実行しないようにするための対策としては、バインド機構の使用及びエスケープ処理がある。」と認定されています。SQLインジェクション攻撃は、情報システムの脆弱性を狙った代表的な攻撃手法の一つです。

事例3（東京地裁平成26年1月23日判決）

①原告（ユーザ）はインテリア商材の通信販売等を行う株式会社であり、被告（ベンダ）は業務システムの開発、WEBサイトの制作等を行う株式会社である。

②被告は原告から、商品のWEB受注システム（以下「本件システム」という。）の開発について基本契約及び個別契約を締結することで受託し、本件システムを納品した。

③原告が被告から本件システムの引渡しを受けた後、SQLインジェクション攻撃を受けて原告の顧客のクレジットカード情報が流出し、原告に約3200万円の損害が発生したが、過失相殺により原告の損害賠償請求が可能となる金額は約2262万円と判断された。

---

[3]「HeartBleed」という脆弱性の名称は、メモリ上にあるデータが漏洩してしまう問題を、心臓の鼓動によって血液が漏れ出る様子に例えて命名されたようです。

[4] 以下のURL でHeartBleedの技術的な解説がされています。（https://eset-info.canon-its.jp/malware_info/qa/detail/141204_1.html）

[5] IPA「情報セキュリティ 10大脅威2015~被害に遭わないために実施すべき対策は？」24頁参照。

[6] IPA「セキュリティ担当者のための脆弱性対応ガイド~企業情報システムの脆弱性対策~情報セキュリティ早期警戒パートナーシップガイドライン別冊」2015年3月版の7頁参照。

[7] IPAの以下のURLで解説がされています。（https://www.ipa.go.jp/security/announce/struts2_list.html）

第3 脆弱性が残る情報システムを提供したベンダの法律上の責任

上記のように、情報システムの脆弱性を攻撃された結果、当該システムを利用するユーザ企業に損害が発生するという事案が多数発生しています。脆弱性攻撃を仕掛けた人物が法的な責任を負うことは当然ですが、脆弱性の残る情報システムやソフトウェアを提供したベンダは、ユーザに対しどのような責任を負うことになるのかを検討してみましょう。

前々回（2017年10月号）の「裁判例から見た情報システムの品質問題」で解説したとおり、ベンダは、不具合について債務不履行責任と不法行為責任を問われる可能性があり、ユーザからは損害賠償請求、瑕疵の修補請求を受け、又、契約を解除される可能性もあります。

そして、「脆弱性」も不具合の一種と見ることができますので、「脆弱性」を攻撃された場合にも、債務不履行や不法行為責任を追及される可能性があることは同じです。

もっとも、「不具合」であるか否かの判定については、ベンダ・ユーザ間で合意した設計書等の内容と、実際に納品した情報システムの動作を対比すれば、ベンダが契約上の義務を負う範囲内の事項であるか否かを判断しやすいという傾向があります。

しかし、「脆弱性」に関しては、設計書等にその内容が明記されていないことが多く、ベンダの債務が十分に特定されていないプロジェクトも多いように思います。

このような場合にどう考えるかという点について、東京地裁平成26年1月23日判決は、以下のとおり判示しています。

脆弱性とベンダの債務に関する東京地裁平成26年1月23日判決の判断

被告（ベンダ）は、平成二一年二月四日に本件システム発注契約を締結して本件システムの発注を受けたのであるから、その当時の技術水準に沿ったセキュリティ対策を施したプログラムを提供することが黙示的に合意されていたと認められる。そして、本件システムでは、金種指定詳細化以前にも、顧客の個人情報を本件データベースに保存する設定となっていたことからすれば、被告は、当該個人情報の漏洩を防ぐために必要なセキュリティ対策を施したプログラムを提供すべき債務を負っていたと解すべきである。

ポイントは、「当時の技術水準に沿ったセキュリティ対策

を施したプログラムを提供することが黙示的に合意されていた」と判断されている点です。ベンダとしては、契約書等でセキュリティ対策について何も触れられていなくても、契約を締結した当時の技術水準に沿ったセキュリティ対策を実施する義務を負う可能性があり、この義務に違反した場合には、債務不履行や不法行為を理由に損害賠償請求を受ける可能性があることになります。

例えば、SQLインジェクションやクロスサイト・スクリプティングのように、脆弱性としての届け出数も多く[8]、IPAが脆弱性の対処方法まで公開しているような場合に[9]脆弱性対策を実施していないと、「当時の技術水準に沿ったセキュリティ対策を施していない」との判断になる可能性があります。

第4 契約書等で規定されている損害賠償請求を免責・制限する条項の適用

上記のように、ベンダはユーザから債務不履行や不法行為に基づく損害賠償請求を受ける可能性があり、しかも賠償額が高額となる可能性もあることから、損害賠償義務を免責する条項（以下「免責条項」といいます。）や損害賠償義務を

[8] IPA「ソフトウェア等の脆弱性関連情報に関する届出状況［2017年第1四半紀（1月～3月）］」の13頁では、過去2年間で届出された脆弱性のうち、クロスサイト・スクリプティングが約56%、SQLインジェクションが約11%を占めていると報告されている。

[9] IPA「安全なウェブサイトの作り方」（改訂第7版）では、SQLインジェクション、クロスサイト・スクリプティングを含む11種類の脆弱性について対策を解説している。

負うとしても、損害賠償額に上限を設けるという条項（以下「責任制限条項」といいます。）を、契約に盛り込むことがあります。しかし、免責条項等を契約書に盛り込んでおけば安心であるという考え方は危険です。

この点について、前述の東京地裁平成26年1月23日判決は、以下のとおり判示しました。

責任制限条項の適用に関する東京地裁
平成26年1月23日判決の判断

①ソフトウェア開発に関連して生じる損害額は多額に上るおそれがあることから、被告（ベンダ）が原告（ユーザ）に対して負うべき損害賠償金額を個別契約に定める契約金額の範囲内に制限すること自体は合理性がある。

②被告が、権利・法益侵害の結果について故意を有する場合や重過失がある場合（その結果についての予見が可能かつ容易であり、その結果の回避も可能かつ容易であるといった故意に準ずる場合）にまで同条項によって被告の損害賠償義務の範囲が制限されるとすることは、著しく衡平を害するものであって、当事者の通常の意思に合致しない。

③本件基本契約二九条二項（責任制限条項）は、被告に故意又は重過失がある場合には適用されない。

要するに、上記①のように、ベンダが免責条項や責任制限条項を設けることについての合理性は認めつつも、権利・法益侵害の結果について、ベンダに故意又は重過失がある場合にまで責任が制限されるのは、著しく衡平を害するとして、結局、ベンダの故意又は重過失の有無によって、免責条項や責任制限条項が適用されるか否かを判断していると言えます。

ここで言う「故意」とは、「わざと」「意図的に」という意味ですが、「重過失」とは、どのような場合を意味するのでしょうか。

前述のとおり、同判決では「重過失」について「結果についての予見が可能かつ容易であり、その結果の回避も可能かつ容易であるといった故意に準ずる場合」という基準を用いています。一般的に「過失」とは、権利・法益侵害の結果について、①予見可能性があり、②結果を回避できる可能性がある場合に認められます。これに対し、「重過失」の場合は、①予見可能性があるだけではなく、容易に予見できることが必要であるとし、②結果を回避できる可能性があるだけではなく、容易に結果を回避できることまで必要であるとしています。

東京地裁平成26年1月23日判決では、上記の基準を採用した上で、以下のとおり、ベンダの重過失を認めています。

重過失の判断基準に関する東京地裁
平成26年1月23日判決の判断

①被告（ベンダ）は、情報処理システムの企画、ホームページの制作、業務システムの開発等を行う会社として、プログラムに関する専門的知見を活用した事業を展開し、その事業の一環として本件ウェブアプリケーションを提供しており、原告（ユーザ）もその専門的知見を信頼して本件システム発注契約を締結したと推認でき、被告に求められる注意義務の程度は比較的高度なものと認められる。

②SQLインジェクション対策がされていなければ、第三者がSQLインジェクション攻撃を行うことで本件データベースから個人情報が流出する事態が生じ得ることは被告において予見が可能であり、かつ、経済産業省及びIPAが、ウェブアプリケーションに対する代表的な攻撃手法としてSQLインジェクション攻撃を挙げ、バインド機構の使用又はSQL文を構成する全ての変数に対するエスケープ処理を行うこと等のSQLインジェクション対策をするように注意喚起していたことからすれば、その事態が生じ得ることを予見することは容易であったといえる。

③バインド機構の使用又はエスケープ処理を行うことで、本件流出という結果が回避できたところ、本件ウェブアプリケーションの全体にバインド機構の使用又はエスケープ処理を行うことに多大な労力や費用がかかることをうかがわせる証拠はなく、本件流出という結果を回避することは容易であったといえる。

この事例においては、IPAがウェブアプリケーションに対する代表的な攻撃手法としてSQLインジェクション攻撃を挙げ、対処方法を示して注意喚起をしていたことから、対策をしなければ情報漏洩等の結果を招くことを予見することは容易であり、かつ、適切に対処していれば、情報漏洩等の結果を回避することも容易であったとの判断になり、重過失が認められました。

第5 ベンダが脆弱性問題を回避するための工夫

脆弱性といっても、①契約締結時に既知となっている脆弱性、②契約締結後、引渡しまでに既知となった脆弱性、③引渡し後に既知となった脆弱性について、それぞれ対処方法を変えるべきではないかと思われますので、以下のとおり3つの場合に分けて検討してみます。

1.契約締結時に既知となっている脆弱性について

契約締結時に既知となっている脆弱性については、原則として対応しておくべきでしょう。特に、SQLインジェクションやクロスサイト・スクリプティングのように著名な脆弱性で、対処方法も公開されているような脆弱性については、東京地裁平成26年1月23日判決を前提とすると、これに対応すべき法律上の義務があると言え、仮に、免責条項や責任制限条項にユーザが合意していたとしても、裁判においては、ベンダに重過失があると判断されてしまう可能性が高いでしょう。

もっとも、脆弱性対応を完璧に対応しようとすると、その分、コストもかかることになりますから、予算の枠内で対応できない場合もあるかも知れません。このような場合には、対応すべき脆弱性に優先順位を付け、優先度の低い脆弱性については対象外とすることを、ユーザに合意していただくというのが理想的ではないかと思います。

2.契約締結後、引渡しまでに既知となった脆弱性について

では、契約締結時点では発見されていなかった新規の脆弱性が新たに発見された場合はどうでしょうか。ベンダは新規の脆弱性に対応することを前提に契約を締結していないため、新規の脆弱性への対応は契約対象外の業務となり、一定の作業量を超える場合には、無償の対応はできないということになりそうです。しかし、ユーザの立場からすると、新規の脆弱性に対応するため高額な費用を支払ってほしいとの依頼を受けても、予算を確保できるとは限りません。

従って、このような場合は、ベンダがユーザに対し、新規の脆弱性が発見された場合にどのような対応をすることになるのか、事前に説明しておく必要があるでしょう。具体的には、新規の脆弱性に対応するための委託料の増額や開発期間の延長の可能性について、提案書等で説明しておくべきではないかと考えられます。

3.引渡し後に既知となった脆弱性について

情報システムの開発が完了し、運用・保守のステージに入った後に発見された新規の脆弱性についても対応する必要性はありますが、対応するためにはコストや時間がかかります。

従って、運用・保守段階で発見された新規の脆弱性については、開発に関する契約とは別に運用・保守契約の中で対応することとし、その際の報酬の決定方法についても、説明しておくことが望ましいでしょう。

第6 まとめ

開発部門の方は、ユーザから要求された機能の実現を優先させ、脆弱性対応に関する検討が後回しになってしまいがちです。

しかし、脆弱性問題は、しっかりと対応をしておかないと情報漏洩やウェブサイトの改ざん等の深刻な損害を発生させる問題であり、対応をするためには委託料を増額せざるを得ない場合や、納期を変更せざるを得ないほどの工数を要する可能性があります。

従って、ベンダとしては、提案段階等において既知の脆弱性についての対応範囲を定めるとともに、ユーザに対し、開発中や納品後に新規の脆弱性が発見された場合のリスクを十分説明した上で契約を締結することが必要であると考えられます。

以上

イベント参加レポート

当社はソフトウェア検証分野でさまざまなソリューションやサービスを皆様にご提供しています。
より多くの方に当社のサービスを知っていただきたく、各種イベントやセミナーに参加しています。

第10回 国際カーエレクトロニクス技術展

2018年1月17日～19日
東京ビッグサイト

カーエレクトロニクスの進化を支える技術が一堂に出展する国内最大級の専門展「国際カーエレクトロニクス技術展」に昨年に引き続き出展し、以下のサービスをブースにてご紹介しました。

テーマ：
『検証で挑む！カーエレクトロニクスの信頼性向上』

出展内容：
「自動運転／ ADAS ソリューション」
「コネクティッドソリューション」
「品質ソリューション」

ソフトウェアテストシンポジウム 2018 東京（JaSST'18 Tokyo）

2018年3月7日～8日
日本大学理工学部駿河台校舎1号館

２日間にわたり、ソフトウェアテストやプロセス品質について当社社員が講演及びディスカッションを行いました。また、当社の最新ツールをブースにてご紹介しました。

第２回【関西】組込みシステム開発技術展（ESEC）

2018年2月21日～23日
インテックス大阪

イベント・セミナーの情報は
以下からご覧いただけます
http://www.veriserve.co.jp/event_seminar/

現新比較検証サービスのご紹介

コンピュータシステムのベンダー及びユーザー企業において現新比較テストへの関心が高まりを見せています。今回は現新比較テストの問題点と課題を挙げ、効率的な現新比較テストの実施を支援すべく当社が昨年よりサービスを開始した「現新比較検証サービス」をご紹介いたします。

1. はじめに

1990年代から2000年代前半に構築された企業の情報システムの多くが新技術の登場により老朽化の問題に直面しています。長年の運用により複数のサブシステムが複雑に連携するケースが多く、システム改修に要するコストは増加の一途をたどっています。

コスト削減策のひとつとしてリエンジニアリング等の手法も進められていますが、システム改修による影響範囲を絞ることが難しくテスト量も膨大になっています。

また、システム改修の際に、現行機能保障の観点から実施する現新比較テストの進め方についても課題はあります。テスト担当者が目視で主観的な比較をした結果合否判定の基準が曖昧になり、比較結果の証跡も残りにくいという問題です。さらに担当SEが現新比較テストにかかりきりになることで、開発業務に専念できないという問題もあります。

これらの問題解決のため、当社は住友セメントシステム開発株式会社（以下、スミテム）様と提携し、「現新比較検証サービス」の提供を開始しました。

2. 現新比較検証サービスの特長

「現新比較検証サービス」では、スミテム様の画像比較に強みを持つツール「CP-ACCEL」を活用することで画像比較を通じた現新比較テストを行います（図1、図2）。

画像比較テストの対象は基幹系システムや業務系システムの画面や帳票などです。

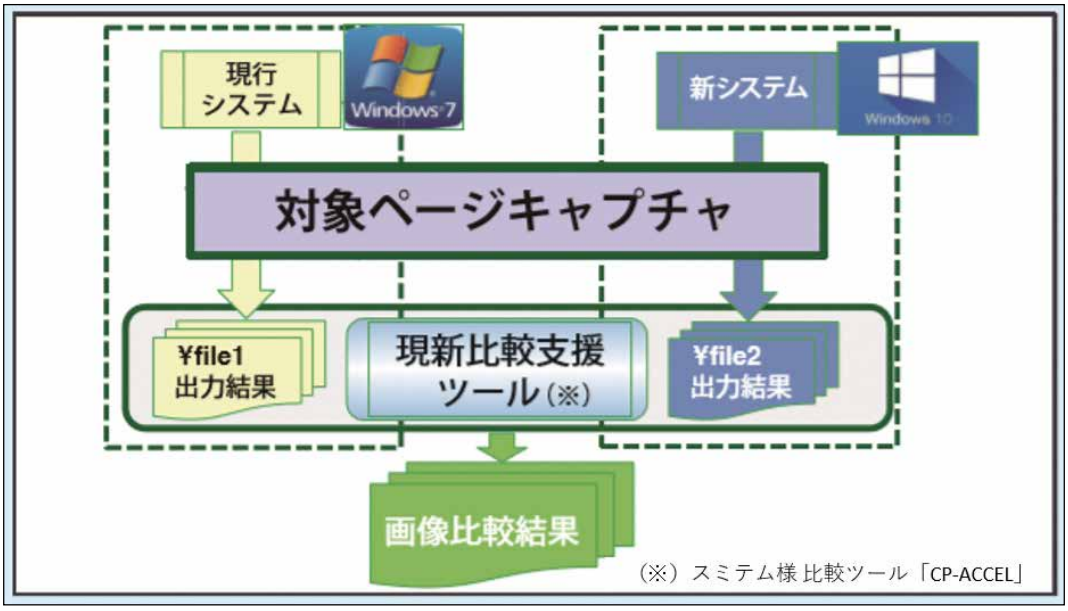


図1

図2

主な特長は以下の通りです。

(1) ツール活用による比較作業の効率化とノウハウの可視化

スミテム様の画像/テキスト比較ツール「CP-ACCEL」の活用により、比較作業の効率化およびノウハウの可視化を実現します。

- 比較結果やツールの設定情報などをテスト結果の証跡として残すことができます。
- テスト結果や証跡は次回テストを行う際の基礎情報やテストノウハウとして活用できますので、効率的な繰り返しテストを行うことができます。

(2) プロセスの標準化

- ノウハウとして蓄積されたテスト結果を活用することで、初めて現新比較テストを行うエンジニアでもすぐにテストを実施できます。
- テスト結果の蓄積により、テスト対象のシステムを改修する際に影響が出やすい箇所やテストの勘所を整理できます。

3. 現新比較テストの需要とご提案

現新比較検証サービスは、特定の業種や業態に縛られることなく様々なお客様に活用頂けるソリューションです。例えば、金融業界における決済フロント画面の改修時の現新比較テストや製造業における社名変更に伴うライン帳票の現新比較といったニーズがございます。

またWindows7からWindows10へのOS移行に関連した現新比較や、Windows10のテクニカルバージョンアップごとの現新比較に対する需要がございます。その他、システムマイグレーション時の回帰テストやサーバリプレイス時の現新比較テストにも活用頂けます。

一般的に画像比較を中心とした現新比較テストでは対象となるシステムの画面数が多ければ多いほどテスト項目も増加する傾向にあります。当社ではシステムのユーザーの利用傾向からテスト割合を定めることでテスト項目を削減させるといったご提案をしております。また利用頻度に応じたパターンをヒアリングしワークフローを作成することやテスト項目書をテンプレート化することで、優先順位に応じたテストの実施により効率的なテストが実施できるよう支援しております。

こうした取り組みを通じて、システムに詳しいエンジニアの手を煩わすことなく経験の浅いメンバーであっても現新比較テストを実施できる環境を実現しております。

また効率的に現新比較テストを行うための手法としてミラーテストがありますが、当社では各種ツールを活用してミラーテストに関する提案も行っております。基幹系システムのオープン化や受発注系システムなど大量の画面比較が必要なケースにおいても効率的なテストに向けたご提案を行います。

4. 今後の現新比較検証サービス

今後、現新比較テストをさらに効率的に行うことができるように、テスト仕様書への合否判定の書き戻し機能、比較精度のカスタマイズなどを進めております。また、エンジニアによる機能確認業務との組み合わせや、当社が提供しているテスト自動化ソリューションと組み合わせることなどによりテストの生産性向上を図る予定です。

これらの機能拡張時には別途当社ホームページにご案内させていただきます。

現新比較検証サービスにご興味がありましたら、以下までご連絡をお願いいたします。

株式会社ベリサーブ　担当窓口　Tel：**03-5909-5702**（田中、御木）

VERISERVE NAVIGATION（ベリサーブナビゲーション）
2018年3月号

編集・発行
株式会社ベリサーブ
東京都新宿区西新宿6-24-1 西新宿三井ビル14F

お問い合わせ
マーケティング部：03-6302-0707
verinavi@veriserve.co.jp