WHITE PAPER | 2016 年 3 月

DevOps とテスト担当者

テストの専門家でありコンサルタントであるPaul　Gerrard氏は、テストに関する幅広いトピックについて一連の記事を著しています。本書ではテスト担当者とテストという観点からDevOpsの採用について説明しています。DevOpsは高品質なソフトウェアの頻繁なデリバリのために企業が使用する全体的なアプローチの一部です。DevOps実装が成功した場合に得られる最も明確な成果は、ソフトウェアの変更がアイデアから本番へと移行するのにかかる時間が短縮されることです。

Paul Gerrard
Gerrard Consulting

Sponsored by
ca technologies

テスト担当者にとっての DevOps の意味

背景

この記事ではテスト担当者とテストという観点から DevOps の採用についてお話しします。DevOps ムーブメント（他に適切な名前がありません）は急速に進行しています。業界で起きたほかの多くのムーブメントと同じく、採用のスピードはムーブメントそのものの定義を超えて加速しています。DevOps はいまだに適切に定義されておらず、文化の微妙な違い、新しいテクノロジから出現した機能、ケース･スタディの範囲（大部分は成功例）などを見ると、目の前にある課題がいまだに広く議論されていることがわかります。[1]

人によって、DevOps を問題への解決ととらえる場合も、それ自体が目標の場合もあります。企業によっては目標は「デジタル化すること」であり、DevOps は高品質なソフトウェアの頻繁なデリバリへの総合的なアプローチの一部です。本書ではこのような文脈を前提としてお話します。ただし DevOps 関連のテクノロジおよびサービスのマーケティングでは、この目標は曖昧なことがあります。成功に必要な文化的変化（具体的に言えば動作の変化）という課題は、多くの場合、過小評価されています。

もう 1 つ前提とするのは、DevOps に関与しそれに影響を受けるテスト担当者が、DevOps の考え方になじみがないということです。本書はこのようなテスト担当者への入門書となることをめざし、また、テスト・プラクティスに DevOps が与えるインパクトについても議論します。本書は DevOps の経験が豊富な実践者の方にも有益なものになるでしょう。テスト担当者の方であれば、少なくともテスト担当者の観点を確認できます。

初心者の方へ：DevOps とは？

簡単に言えば DevOps とは、緊密に連携する開発チームとシステム運用チームという概念を表す名称です。ソース･コードのコミットから本番での運用まで、いわゆるデリバリ・パイプラインにおいて、開発者は運用アクティビティのいくつかを引き受け自動化します。運用チームは開発者のアクティビティに対してもっと可視性を備えており、いくらかの影響も与えます。DevOps 導入の動機は主に、ソフトウェアのデプロイと実装を迅速化することです。運用と開発の連携を向上させ、効果的に 1 つのアジャイル・チームを作ることで、「アジャイル運用」と呼べるものを実現できます。

DevOps の成功の実装の最も明確な成果は、ソフトウェアの変更がアイデアから本番の運用へ進むために要する時間の短縮です。ソフトウェアの変更が「完了した」と開発者が言うとき、本番での使用への移行は広範な自動化の助けによって実行されています。自動化されたツールやプロセスはシステム構成やビルド・プロセス、テスト、テストやステージング、本番環境へのデプロイ、デプロイ後の監視、評価、運用で使用されています。

では、DevOps とはツールのことでしょうか？

1 つのレベルでは、DevOps の目標は自動化によってデリバリ・パイプラインのボトルネックを排除することです。ただし段階的なプロセスの自動化はまだガバナンスを必要とします。最も自動化されたプロセスもそれほど自律的ではありません。保守や例外処理では人間の介在なしにタスクを完了することはできません。人的要因の考慮なしには、完全に自動化された DevOps プロセスも意味がありません。ツールは手間のかかる作業の多くを担いますが、ツールを正常に動作させる（またはそれに失敗する）プロセスを実行するのは人間です。

では、DevOps とは開発と運用のスタッフがツールの力を借りて緊密に作業するということでしょうか？

いいえ、それも違います。自動プロセス間の引継ぎの多くは他のプロセス（通常は何らかのテスト）を必要とします。自動テストは開発者とテスト担当者が構築する必要があります。こうしたテストの結果は、パイプラインのステージ間移動のために他のプロセスやスタッフに十分な情報を提供することに焦点が当てられます。テストを行うテスト担当者と開発者は、DevOps プロセスが首尾よく確実に達成されるという保証を与えます。

「よくわかりません。DevOps とは本当は何ですか ?」それは発展しつつある新しい規律です。ある優れた記事に、問題が提示され詳細に議論されています。[1] この議論が起きたのは本書を書くわずか数週間前のことです。つまり DevOps の定義はまだ定まっていないということがわかります。おそらく定まることはないでしょう。

これはテスト担当者にとって何を意味するでしょうか ?DevOps に関して「1 つの正しい方法」はまだないということ、そして進化している DevOps 体制（あらゆる体制は進化しています）におけるテスト担当者の役割はまだ定まっていないということです。テスト担当者が行える主な貢献は以下の 2 つです。

1. 手間のかかる作業に注目し、手間を減らすよう取り組む必要があります。
2. DevOps プロセスに価値を付加する機会と介在について特定する必要があります。

DevOps に向けた推進要因を最もよく表す合言葉があるとすれば、それは「大変なことは何度も実行すべき」です。これは少し陳腐な表現かもしれませんが、本書では DevOps のテスト・プラクティスを実装し改善するためのコンテキストとして、これを使用します。

大変なことは何度も実行すべき

特定の仕事をするときに経験する困難や手間は、マイナスに影響します。作業をやりたくないときは、後回しにしがちです。最終的に取り掛かったときには、もっと面倒なことになります。これは歯医者に行くときでも、ガレージの掃除やソフトウェアの統合、テストの実施でも同じです。こうした作業は行う頻度が低いほど、実際にそれを行うときに苦痛が増すことを私たちは経験しています。特定の作業を頻繁に行ったり継続的に行ったりすると、なぜ負担感を減らせるのかについて、Martin Fowler 氏は 3 つの理由を挙げています。

第 1 の理由は、大規模で複雑な作業は計画や管理、制御を行うのが困難であるからです。大規模な作業を分割することで実行が容易になり、リスクも減り、うまくいかないことがあっても簡単にやり直せます。第 2 の理由は、作業を数多く行うとフィードバックが得られるからです（テストは顕著な例です）。このフィードバックは早期に頻繁に受け取れば、問題に迅速に確実に対応でき、それ以上の時間を無駄にせずに済みます。第 3 の理由は、アクティビティをより頻繁に行えば上達するからです。私たちは何度も行うことで効果的に方法を習得します。また、何らかの方法でそれを自動化できる機会について見極めることもできます。

テスト担当者の観点から見ると、この方針に従えばテスト・プロセスの自動化という考え方をもっとずっと真剣にとらえられるようになります。手作業による介在（通常、DevOps のプロセスにおける自動プロセスと自動プロセスの間に）がある場合、それは手間、つまりプロセスのボトルネックや遅延の原因、プロセスの信頼性が低くエラーが起きやすい側面とみなされます。手作業のテストは手間がかかります。そのとおり、テスト担当者は予備的なテストを好むかもしれません。自動化によって見つけられないバグを見つけられるのは人間だけであり、大きな障害が起きるのを防止するのに信頼できる人間はテスト担当者だけだという懸念を持っているかもしれません。

テスト担当者にとっては、開発者と自動化がテスト作業を適切に行えると信頼することは、苦痛であるかもしれません。もしそれが苦痛なら、それをもっと頻繁に行う必要があるのです。

テスト、自動化、信頼

たとえば確認やテストの意味については多くの議論があり、[3] テスト担当者に対する信頼、確認と自動化に対する信頼に関しても多くの議論があります。[4, 5]

自動確認がすべて信頼できると言いたいわけではありません。明らかにそれよりもっと高度な考え方が必要です。ただし本書では、少なくともテストとテストの実行アクティビティを以下の 4 つのコンポーネントに分割できます。

1. コンポーネント・レベルのチェックインと継続的インテグレーション・プロセスの一部として、開発者によって自動化できる確認
2. API レベルのリンクまたはエンドツーエンド・トランザクションを実行するために（通常システム・テスト担当者によって）自動化できる確認
3. ブラウザ、オペレーティング・システム。プラットフォームにまたがる互換性を証明するために、互換性確認を実行できるテスト
4. 人間によってのみ実行できるテスト

これらを区別する方法について私は 2、3 の提案を行うことしかできません。すべての環境は異なるからです。もっと本書に即した問いは、「どのようにしてテスト担当者は後期の手作業による確認をやめられるか」ということです。以前の記事に書きましたが、後期の手作業による確認を排除するにはプロアクティブな取り組みと信頼が必要です。[6]

以下は、焦点を当てるべき主な取り組みです。

1. 手作業による確認でコンポーネント・レベルでの実行が可能なものは、可能な限り開発者へ渡すべきです。テスト担当者はペアリングまたはホワイトボード・セッションでこれらのテストを提案できるでしょう。テスト担当者は自身でそれらを記述し、継続的インテグレーションの体制にそれらを含める必要があるかもしれません。
2. エンドツーエンドまたはユーザ・インタフェースのテストは自動化を必要とする可能性があります。これらは実行に時間がかかり、脆弱で、頻繁な保守を要するため、最小化する必要があります。これらをすべてのコードのチェックインで実行する必要があるか、より規模が大きく頻度が少ないリリースで使用するために取っておくかを検討します。
3. リリース候補に統合されていないコンポーネントには、手作業のみで行うテストのどれを実行できますか？手作業によるテストは開発者とのペアリング・セッションで実行できますか？このテストの代わりとなるものはありますか？ストーリーボーディングや BDD スタイルのプロトタイプは役に立ちますか ?UI の確認はモックアップまたはワイヤーフレームで実行できますか？
4. 回帰目的で保持する必要がある確認と異なり、1 度だけ手作業で実行する必要がある、自動化の候補となる確認はどれですか？

先ほど、信頼の概念について説明しました。これを別の観点から見ると、後期の手作業によるテストがまったくない場合、システムを確実にテストするにはどうすべきかについて考慮することになります。すべてのテストがツールによって実行される環境を想像してみてください。あなたが抱く懸念の多くを占めているのは、開発者がテストをうまく実行できるとは信じられないということでしょうか？テスト思考を前倒しすれば（著者の以前の記事で提案したように）、疑いを減らすはずです。テスト担当者がリスクを特定して評価する先導者としての役割を務め、テストを選択し、それらが開発と自動化に組み込まれるように行動するなら、テスト担当者の懸念は最小化できます。

確かに、テスト担当者は自分こそが品質を守るゲートキーパーであり、最後のとりであり、気にかけているのは自分だけだと信じることをやめる必要があります。テスト担当者はもっとビジョナリーやリスクの特定者、リスク管理者、パスファインダー、ファシリテータ、コーチやメンターのような考え方をする必要があります。

プラクティス、監視、改善

善意で後期の手作業による確認への依存を減らしたり排除したりしようとしても、バグは残ります。問題はソフトウェアが本番にリリースされるときに発生します。運用の観点から見た DevOps の重要な規律の 1 つは、より深いレベルの監視です。

コンポーネントからアプリケーションの単純なトランザクション、統合、メッセージング、インフラストラクチャそのものに至るまで、すべてのレイヤでの監視です。監視の目標の 1 つは、ユーザが問題のインパクトを経験する前に障害に対するアラートを発することです。これはかなり野心的な目標ですが、これが最終的な目標です。

本番で問題に遭遇したときのタスクは、監視から引き出した分析を使用し、原因を追跡して解決するだけでなく、自動であれ手作業であれテスト・プロセスを改良し、将来同様の問題が発生する可能性を低減することです。パイプラインのプロセス全体にわたるテストと分析の役割は、ここで導入され議論されました。[7]

DevOps プロセスにおける自動テストは「監視」と呼ぶことができます。本番での監視と組み合わせれば、DevOps プロセス全体と本番への監視はテストの適用範囲を拡大すると言えるでしょう。つまり DevOps はテスト担当者の役割を減らすわけではないのです。

まとめ

最近私は「企業が DevOps を試みるべきではないのはどのような場合ですか ?」と尋ねられました。良い質問ですが、この背景にあるのは DevOps が普及するかどうか、そしてテスト担当者は気にとめるべきかどうかに対する懸念でしょう。その答えは簡単です。

開発者と運用スタッフが会話することを望みませんか ? テストと本番にもっと信頼できるビルドとデプロイが欲しいと思いませんか ? もっと正確で効率的な、情報を得られるパイプラインをサポートする最高のテクノロジが欲しいと思いませんか ?DevOps は素晴らしいことですが、実現するのは必ずしも簡単ではありません。言うまでもなく、文化的変化を必要とし、それは簡単だとは限りません。

テスト担当者にとって DevOps はプロジェクトの初期段階により大きな影響を及ぼします。また、テストの自動化、情報のプロビジョニング、意思決定に関してもっと真剣に考えさせるものです。テスト担当者は DevOps を受け入れる必要があります。プロアクティブになり、プロジェクト・チーム内でさらに権限と尊敬を得られる機会を DevOps は提供するからです。

著者について

Paul Gerrard 氏はコンサルタントであり、教師、執筆者、Web マスター、開発者、テスト担当者、カンファレンス講演者、コーチ、パブリッシャーでもあります。テスト保証に特化したソフトウェア・テストおよび品質保証のあらゆる側面でコンサルティングを担当してきました。欧州、米国、オーストラリア、南アフリカにわたるテスト･カンファレンスで基調講演や個別指導を行い、その一部は賞を受賞しています。

オックスフォード大学と Imperial College London で教育を受け、2010 年には Eurostar European Testing の優秀賞を、2013 年には European Software Testing Awards（TESTA）の特別功労賞を受賞しました。

2002 年に Neil Thompson 氏と共に『Risk-Based E-Business Testing』を著しました。2009 年に『The Tester's Pocketbook』を著しました。2011 年に Susan Windsor 氏と共同で『The Business Story Pocketbook』を執筆し、2014 年に『Lean Pytho』を著しました。

2014 年にはダブリンで開催された EuroSTAR Conference のプログラム議長を務めました。

Gerrard Consulting Limited の社長であり、TestOpera Limited の役員、Test Management Forum の司会者も努めています。

メール : paul@gerrardconsulting.com
Twitter : @paul_gerrard
Web : gerrardconsulting.com

詳細については CA Technologies の**開発とテスト**を参照してください。

ca.com/jp/でCA Technologiesにアクセスしてください。

CA Technologies（NASDAQ：CA）は、企業の変革を推進するソフトウェアを作成し、アプリケーション・エコノミーにおいて企業がビジネス・チャンスを獲得できるよう支援します。ソフトウェアはあらゆる業界であらゆるビジネスの中核を担っています。プランニングから開発、管理、セキュリティまで、CA は世界中の企業と協力し、モバイル、プライベート・クラウドやパブリック・クラウド、分散環境、メインフレーム環境にわたって、人々の生活やビジネス、コミュニケーションの方法に変化をもたらしています。詳細については **ca.com/jp** をご覧ください。

参考資料

1. 「What is DevOps」、The Agile Admin、http://theagileadmin.com/what-is-devops/
2. 「Frequency Reduces Difficulty」、Martin Fowler、http://martinfowler.com/bliki/FrequencyReducesDifficulty.html
3. 「Testing and Checking Refined」、James Bach、Michael Bolton、http://www.satisfice.com/blog/archives/856
4. 「A New Model for Testing」、Paul Gerrard、http://dev.sp.qa/download/newModel
5. 「The New Model and Testing v Checking」、Paul Gerrard、http://blog.gerrardconsulting.com/?q=node/659
6. 「How to Eliminate Manual Feature Checking」、Paul Gerrard、ウェビナー、http://blog.gerrardconsulting.com/?q=node/622
7. 「Thinking Big:Introducting Test Analytics」、Paul Gerrard、http://blog.gerrardconsulting.com/?q=node/630

CS200-181159_0316