テストマネジメント虎の巻　第五回

日本ヒューレット・パッカード株式会社
HP ソフトウェア事業統括　湯本剛

計画編その 3　～スケジューリング（やることの具体化）～

皆さんこんにちは。今回のテストマネジメント虎の巻は計画編の続きです。前回までは計画のなかで考えるべき事を3つ説明しました。今回からは見積もりについての解説と前回に予告しましたが、計画の話の中で解説すべき事がいくつか残っているので、先にそちらから解説する事にします。今回は「テストのやり方を決める」部分の具体化、つまりスケジュールの策定をとりあげます。

テストのために検討した事を元にスケジューリングする

前回の本連載では、計画書を作る前に行うこと（テストのために検討すること）としては、大きく次の 3 つがあるとしました。

- テストする範囲を決める
- テストする環境を決める
- テストのやり方を決める

上記 3 つの検討が進んだら、テストのやり方を更に具体的にするために、自分が受け持つこととなったテストの日程、つまりリリースまでにどのようなテストをどんな順番でしていくかを決めていきます。テストをするといっても、思いつくがままにどんどんテストをするのではなく、どういう順番でテストをやるのが効率よいのか？といったことや、どうやればバグが早めに見つかるのか？と言ったことを考えていかないとテストは上手く進まないからです。

テストの日程を決めると言っても、「テストケースも作っていないのに日程なんて考えられない」と思う人もいるかもしれません。日々のやることを細かく日程にしようとすると確かにその通りかもしれません。ただ、テストケースができるまで何も日程を考えないわけにはいきません。そこでどうするかというと、テストすべき事を「新機能」とか「既存機能」といった大きなグループとしてとらえて、無駄が少ないテストの進め方や重大なバグが早く見つかる進め方を考えながらそのグループをリリースまでの日程の中に配置していきます。映画の 1 シーンであるような、作戦会議にて地図を広げて駒の配置を考える様子をイメージしてもらえればよいかと思います。

リリースまでの日程の中に配置した「テストすべき事のグループ」を HP では「サイクル」と呼びます。同一アプリケーションだとしても、サイクルは常に同じではありません。新規開発とバージョンアップ開発ではテストすべき事が異なりますし、バージョンアップでも機能の追加度合いや変更度合いによってテストすべき事が異なります。そのため、図 2 のように、各リリースにはそのリリースに適したサイクルを定義します。

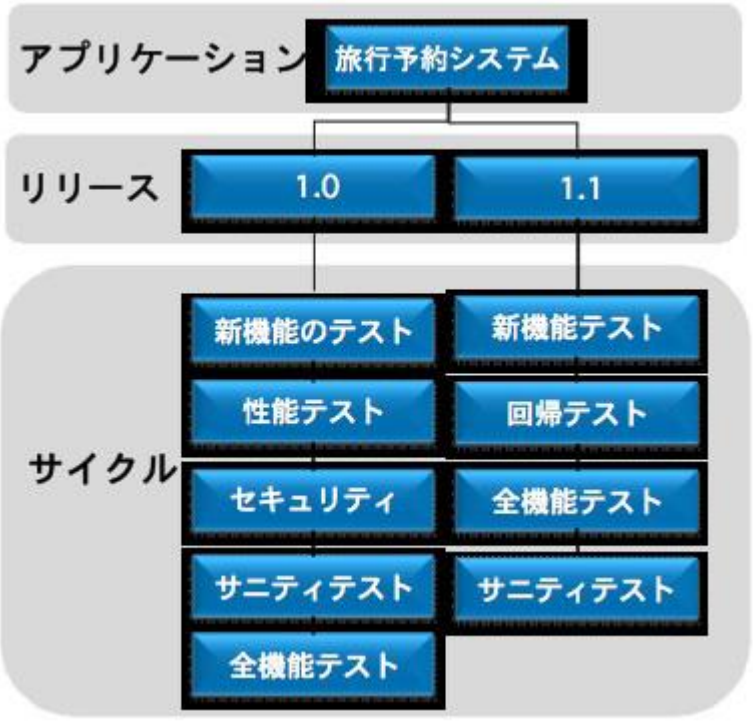

図 2 リリースとサイクルの関係

テスト対象のアプリケーションをリリースするまでにサイクルをどう配置してテストしていくかを決める事が「スケジューリング」になります。このレベルのスケジューリング、つまり「テストサイクルの配置」は、テストを計画する中でもとても重要な活動です。サイクルの配置を間違えると、テストがリリース日程に間に合わなくなったり、リリースの直前になっても重要な部分がテスト出来ていなかったりといった問題が起こるからです。

サイクルの配置の例

それでは、実際にどのようにサイクルを配置するかを、とある旅行会社の旅行予約システムのバージョンアップ案件（バージョン 1.1）を例に考えてみましょう。

このシステムでは、旅行先でちょっとした観光をするためのオプショナルツアーを選択できると言う機能が追加になったとします。システムテストが可能になったあと、最初は新機能（つまり、オプショナルツアー選択画面）のテストをするのがもっともよいでしょう。なぜなら、新機能は変更の無い既存機能よりバグが多く入っている可能性が高いため、早めにそれらのバグを取り除いたほうがよいからです。

新機能が正しく動く事がテスト出来てから、新機能の影響範囲を選んで回帰テストをやります。旅行予約システムで言えば、予約の合計金額を扱う機能（オプショナルツアーの金額が加算されているか？）や、旅程を印刷する機能（旅程の中にオプショナルツアーが入っているか？）などをテストする事になります。

また、それとは別ラインで、影響が無いと思われる既存機能に対する全体的なテスト（全機能テスト）を行います。この例で言えば飛行機や新幹線などの移動手段の予約やホテルの予約は影響を受けないはずですが、万が一動かないとその影響が多大になるため、念のため基本的なユーザシナリオに基づいて動作させても問題ないことは確認してお

くといった事が該当します。

新機能のテスト、既存機能のテストが大体終了した時点でそこまでのテストで見つかった不具合を分析すると、今回のシステムの不具合の傾向が見えてくると思います。しかし、計画時にイメージした通りに不具合がでるということはめったにありません。そこで、当初用意したテストを補うために、不具合の傾向から考えて更にテストを追加で行うための日程を確保します。(図 3 では「強化テスト」と書いてある部分が該当します。)

リリース直前には、新機能や回帰テストした影響範囲も含めて全機能を一通り動かすテスト(最終動作確認)を行い、不具合が出なければリリース可能になります。

こういったようにテストサイクルを配置していきます。これらの配置はガントチャートのような図に表わしてメンバーで意識合わせをできるようにします。この例をガントチャートで表したものが図 3 になります。図 3 は HP のテストマネジメントツールである Quality Center を使ってガントチャートを書いた例です。

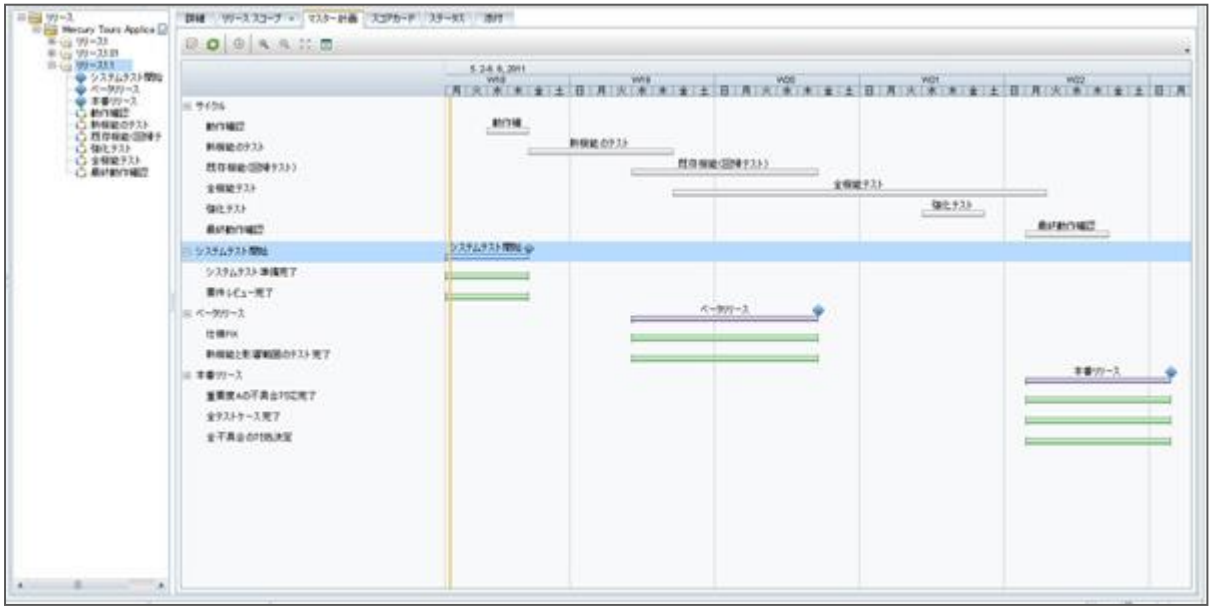

図 3 サイクルの配置とマイルストーンを記載したガントチャート

マイルストーンとの関係を明確にする

また、サイクルを配置してテストを進めていく上で、リリースまでの途中でどのような事が達成できていないといけないか(つまり、マイルストーン)とのつじつまが合っているかを確認します。マイルストーンは、例えば「この日程までにはβ 版としての品質(これは、例えば「主要機能のテストケースの 90%がパスしている」といった具体的な数字にします。)を確保する」とか、「どの日程までには仕様が FIX していること」といった内容が該当します。

システムテスト開始前までに、仕様も全て FIX し、機能も全て実装され、それまでのテストも全て合格しているといったように、開発プロジェクトの作業が順序立てて行われていれば、マイルストーンとサイクルの配置はイコールになるように計画ができます。しかし、いまどきの開発は短納期を実現するために段階的にリリースをしたり、また同時並行的にいろいろな作業が進んでいき期間を短縮したりといったことが行われます。こうなると、サイクルの配置とマイルストーンがイコールにならないため、明示的にマイルストーンの日程を明らかにし、開発プロジェクト全体としての日程とテ

ストの同期が取れるようにします。

マイルストーンも図 3 のようにサイクルと合わせてガントチャート上に図示するとメンバー全員でテストの考え方を共有しやすくできます。

「テストアプローチ」…テストサイクルの配置を考える方法

ここまでの説明だけでは、「テストサイクルの配置を上手くやるためにはどうすればよいのか？」と疑問が出てくる読者もいるかと思います。その場合、例えば以下の様な「テストアプローチ」をベースに考える方法があります。「テストアプローチ」とは、本連載でもたびたび出てくる ISTQB で出てくるテストを上手くやるための考え方です。(http://www.jstqb.jp/syllabus.html) ISTQB のシラバスのテストアプローチを元にいくつか配置の方法の例をあげてみます。

- 分析的アプローチ：品質リスク（市場で不具合が発生してしまう可能性）を分析してその優先度から配置を考える方法
- モデルベースアプローチ：統計的な不具合の発見のされ方（信頼度成長モデル）や一般的なユーザの利用方法（運用プロファイル）といった、過去の実績に基づく統計的な分析で配置を考える方法
- プロセス準拠アプローチ：業界固有の標準的な配置を利用する方法

ユニットテストのような、開発者が自分自身の開発したソフトウェアに対して、自分の意図した通りに動く事をテストする場合は、プロセス準拠アプローチが有効でしょう。なぜなら、ユニットテストはどんなタイプのソフトウェアだとしてもほぼやる事が決まります。つまりコンテキストに依存せずやらなければならない事が決まるからです。

システムテストのように一般的には品質保証を目的にするテストでは、システムの開発の仕方に依存して選択します。アジャイル的な開発では分析的アプローチが有効でしょうし、過去に同様の開発をたくさん経験している場合は、モデルベースアプローチを使う事が有効な場合が多いでしょう。

テスト詳細スケジュール

配置したテストサイクルだけでは具体的ではないため、「このようなスケジュールではテスト実行の進捗を見たりできないのではないか？」と思う人もいるでしょう。それはその通りです。テストサイクルの配置は、日ごとの細かい日程を作っているわけではないからです。テストサイクルの配置はテストのスケジューリングの考え方を理解するのに適していますが、詳細に日程をモニタリングしたりコントロールしたりするには「ざっくり」すぎます。そのため、日程のモニタリングとコントロールができるようにするためには、「詳細スケジュール」を策定します。

つまり、スケジューリングには、「サイクルの配置」と「詳細スケジュール」の 2 段階があるということです。

テスト実行の詳細スケジュールを立てる際には、配置したテストサイクルは、図 4 のように開発したテストケースをサイクルに割り当ていく「箱」のような役割を果たします。テスト計画したガントチャートの日程に合わせて、以降のテスト実行のフェーズの中でテストケースの実施状況や不具合の検出と修正の状況を確認していくことになります。サイクルに割り当てたテストケースの数が具体的な進捗の根拠になるので、テスト実行結果の数でテストが計画通りに進んでいるかを判断する事ができます。ただ、詳細スケジュールはテスト実行の直前まで変更する事もありますし、実行時の

進捗や品質によっても常に見直されていくことになります。そのように見直しが常に入るからこそサイクルを基にしたスケジューリング（つまり、考え方が理解できるレベルの日程表記）が必要になるのです。

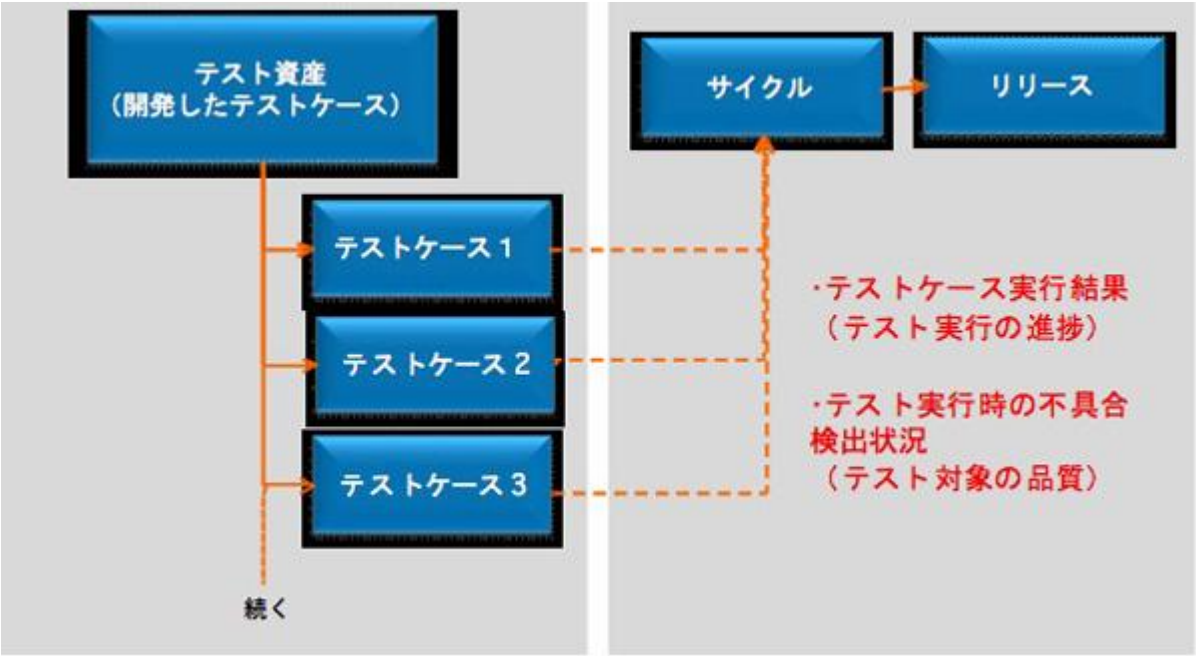

図 4 テストケースのサイクルへの割り当て

今回は「テストのやり方を決める」を更に具体化していく作業として「テストのスケジューリング」について説明しました。次回は、「テスト計画の文書化」についての説明をしたいと思います。