いまさらアジャイル石川
アジャイルプロセス協議会

アジャイルプロセスとウォータフォールの違い

2016/1/23

濱　勝巳@株式会社アッズーリ

アジャイルプロセス（以下、アジャイル）とウォータフォール・プロセス（以下、WF）にはどのような違いがあるのでしょうか？
ドキュメントを作らないことですか？それともイテレーションやスプリントをやるとかやらないとか？はたまた、テストファースでしょうか？継続的なインテグレーションでしょうか？ペアでプログラミングをしていれば良いのでしょうか？それとも残業をしないことですか？
これらは、アジャイルと WF のどちらでも有効には働きます。ですから、これらのプラクティスの実施の有無ではアジャイルと WF の違いをみることはできないと考えています。
例えば、スクラムのプラクティスをすべて実施したとして、１回のスプリントでプロジェクトが終わったとします。これは、アジャイルでしょうか？WF でしょうか？
そう、どちらも違いはありません。
しかし、アジャイルと WF には何かしらの違いがありそうです。それはどこなのでしょう。私は、アジャイルと WF の大きな違いが２つあると考えています。

１つの目違い

まず、１点目は、WF が有期限であることに対して、アジャイルが未期限であるということです。これまで、アジャイルも含めてシステムなどを構築し、オーナーに収めてオーナーがそれを利用していくという考え方が蔓延しています。そのため開発をプロジェクトとして立ち上げ、それが終了した後に、運用へと移すような分け方が一般的に行われています。
2000 年頃に紹介されたアジャイル「開発」宣言もこの開発にのみ注力した「狭義のアジャイル」であったといえるでしょう。
しかし、アジャイルを実践していくと「狭義のアジャイル」ではなく、未期限に運用まで含めて対象を考えていかなければならないということが見えてきました。これを「広義のアジャイル」と呼びます。

プロジェクトは、その定義では有限期間であることが条件になってきていますので、「狭義のアジャイル」はプロジェクト化することができますが、「広義のアジャイル」はプロジェクト化することはできません。
「広義のアジャイル」は、対象のシステムなどを「作ること」が目的ではなく、「使うこと」が目的となります。使う目的のために、対象を作るという手段をとっているにすぎないと考えます。
DevOps などは、アジャイルの本来持っている未期限の発想に注力した考え方であり、開発だけではなく運用を含めた事前準備や取り組みを実践していくことであると考えています。
未期限であるというと、期限を設けてはならないと思いがちですが、これは大きな間違いです。何からのマイルストーンを決めたり、ある期限を設けてそれを目標に進めたりすることはフィードバックを得るためには必要なことです。ここで未期限としているのは、その意識であり、マイルストーンや納品日が来たからといって決して終わりではないということを常に意識して取り組まなければならないということです。
ちなみに、無期限とせずに、あえて未期限としたのは、対象を利用するライフサイクルは必ずあり、どこかで終わりが来るからです。無期限であると永遠に終わりがありませんが、アジャイルは未期限であり、状況に応じてどこかで終わりがくることがあるためあえて未期限としています。
このため、何からのアジャイルのプロセスやプラクティスを実践していたとしても、未期限であることを前提に対象を利用したり、作ったりしていない限り、WF と変わりがないといえます。

２つの目違い

第２点目にあげる違いとして、結果に対して「完全」を前提としているか、「不完全」を前提として取り組めているか、そのような思考でいるか否かということではないかと考えています。どちらが正しいとも間違っているともいうつもりはありません。時と場合にその良し悪しは異なるでしょう。
WF は、「完全」を前提として、システムを実現することを目指しています。神が作ったこの世界には、必ず完璧で完全な到達点があり、それを見出そうと日々努力を重ねようと行動します。西洋的な思考の世界です。完全な状態があれば、お客様のやりたいこを要求や要件を完全に定義して、それを機能的に分割して

組み立てることができれば失敗することはありません。そのため、失敗すると事前の準備が悪かったとか設計が不十分であったとして、完璧な定義や設計をするための手法を模索します。失敗をしないような施策を積み上げていく必要があります。完全な定義や設計ができるのであれば十分に機能する考え方であり、最も効率的に進められることになります。
アジャイルは、不完全を前提としています。世の中は、諸行無常、常がなく変換し続けます。到達したはずの完全もいずれ時とともに不完全に変わっていくと考えます。東洋的な思考の世界です。完全なものは手に入らないので、「とりあえず」を仮定してそこに到達するように進めていきます。その後、時間とともに必ず不完全な状態になるので、それらに対応していく必要があると考えます。アジャイルを知るとこのことはすぐに理解することができます。その上で、設計をしなくて良いとか、仕様書などの文書が必要ないとして、迅速に形にしていくことができるようになります。
しかし、この「とりあえず」が難しい。アジャイルができない人は、この「とりあえず」が決断できないか、「とりあえず」によって継続することができない状況に陥ってしまします。
小さなソフトウェアなどの対象であれば「とりあえず」の判断は比較的容易ですが、例えば、１０年、２０年使うシステムであったり、何億、何十億を超える多額を投資するシステムであったりの「とりあえず」を判断して決める勇気が持てる人がどれだけいるのでしょうか。
責任がない場合であれば身勝手に判断することができますが、責任のある立場でこれを決めることができる人は少ないでしょう。特に、技術面の見識や経験のある人ほど、この「とりあえず」に多くの不安を抱えることになります。この不安があると、どうしても完全なものを追い求めようと思考が働いてアジャイルの思考とは反対に向かってしまいます。
アジャイルが大規模に向かないといわれる原因のひとつに、このこともあると思います。
また反対に、経験や技術力の浅い人の「とりあえず」の多くは、いい加減なもので、セキュリティホールや欠陥を多く含み、あっという間に継続できずにライフサイクルの終焉を迎えてしまうことになります。

有期限の中の WF であれば「とりあえず」ではなく、完全を求めれば良いので

すが、完全がどの時期に見出せるかは不明で、見出せない場合は、そのプロジェクトは失敗に終わることになるでしょう。また、有期限であるため、例え完全が見つかったとしても永遠の完全でない限り、将来も利用し続けることはできません。
そのため、未期限に対象を利用し続けるには、アジャイルに重要な「とりあえず」を判断して進めていかなければなりません。しかし、変化を抱擁した「とりあえず」を決めることが容易でないことは、前にもある通り推測できると思います。この「とりあえず」を手にいれるためには、アーキテクトが必要です。「とりあえず」具現化してしまうと、将来に対応できる可能性が低くなるため、優秀なアーキテクトは、抽象化されたアーキテクチャを決定します。WF は完全を決めてからアーキテクチャを決めるので、アジャイルを実践するには、より有能なアーキテクトの存在が必須となります。

最後に

アジャイルと WF の違いは、プラクティスをやっているか否かではなく、「未期限」と「有期限」であるか、「不完全」と「完全」を前提にして取り組むかという思考の違いです。
ここまで本稿では、アジャイルと WF には２点の違いがあるとしていますが、もしかしたら、「不完全」と「完全」のどちらかを前提とするかということだけしかないのかもしれません。
完全を前提とする場合は、単純に進めることができるため、有期限などの諸条件が揃っている場合には、最適解を出せるこちらのプロセスを採用した方が効率良いプロセスです。
しかし、不完全を前提とする場合は、「とりあえず」を決めて進めていかなければなりませんが、これは誰にでもできることではなく、規模感やライフサイクルによって適切なアーキテクトによるアーキテクチャがなければ未期限を有限にしてしまうことになりかねないことを忠告しておきます。

以上