研究会成果報告書

平成 22 年 4 月 19 日

アジャイル開発 QIMP 研究会

《研究会概要》

名　称	アジャイル開発 QIMP 研究会
期　間	2009 年 10 月～　2010 年 3 月
メンバー	玉川憲（日本アイ・ビー・エム）・ 服部京子（日本アイ・ビー・エム） 大野洋一（日本アイ・ビー・エム）・ 松本尚隆（日本アイ・ビー・エム） 渡辺裕（野村総合研究所）・山中敦（日立製作所） 足達直（日立製作所）・ 神部知明（富士通ソフトウェアテクノロジーズ ） 坂田晶紀（富士通）・ 佐野建樹（日本電気） 長沢智治（マイクロソフト）・ 竹政昭利（オージス総研） 菅野洋史（オージス総研）・久田礼子（オージス総研） 長瀬嘉秀（テクノロジックアート）・設楽秀輔（テクノロジックアート） 町田修一（テクノロジックアート）・ 吉川陽子（テクノロジックアート）事務局 敬称略

《開催日程》

第 1 回研究会：　2009 年 10 月 30 日
第 2 回研究会：　2009 年 12 月 4 日
第 3 回研究会：　2010 年 1 月 14 日
第 4 回研究会：　2010 年 3 月 5 日

《目的》

大規模アジャイルプロジェクトにおける、課題への対応策、考え方を検討した

《検討項目》

1. 品質についての検討
1.1 品質基準について
1.2 設計書の品質について
1.3 品質確認について
1.4 バグ収束曲線について
1.5 テストへのインプットについて
1.6 その他
2. 人材育成についての検討
2.1 アジャイルプロジェクトのプロジェクトマネージャーについて
2.2 開発メンバーについて
2.3 その他のスキル
2.4 作業の粒度
2.5 日本におけるアジャイルエンジニアの育成
3. 契約についての検討
3.1 契約形態（構内請負）について
3.2 受注者、発注者側のリスク
3.3 その他

《研究会成果》

1. 品質についての検討

1.1 品質基準について

・ ウォーターフォールと同様のテストをする。特にリリース時には十分なテストが必要である。
・ユーザーストーリーごとのテストをする必要がある。
・検査部門ではテストの数も重要なので、必要な量のテストを作成する。
・機能単位でのテストが必要である。
・開発工程により、品質の基準は異なっている。
・品質検査時に使うバグ収束曲線は、ウォーターフォールとは異なる。

1.2 設計書の品質について

・設計書はウォーターフォールほど厳密に記述しない。
・動作するソフトウェアをユーザーに見せて、設計書にフィードバックすることもある。
・設計書を詳細に作りすぎると、フィードバックにより、変更が起きたとき、工数がとられる。
・設計書を作らないことはまずないが、もし作らなかったときは、チェック項目を明確にする。

1.3 品質確認について

・品質を確認する仕組みを用意する。メトリクスチェック、カバレッジツール、報告の義務化、設計レビュー、フィードバック、リファクタリング等。複数の方法を組み合わせた方が品質は向上する。
・開発者とテスターをペアにして、品質を向上させる方法もある。
・アーキテクチャ設計の品質については、品質の悪い設計でソフトウェアを作ると、イテレーションを繰り返すことができなくなる。
・定量化できないことは、方法を考えて対応することもできる。

1.4 バグ収束曲線について

・早いフィードバックに対応することで、品質を保証する。フィードバックの件数を数えて指標にする。フィードバックに迅速に対応するには、テスト、静的解析等を自動化する必要がある。
・ウォーターフォールのように、比較できる過去のデータがないため、アジャイル開発の事例が少ないときは、判断が難しい。
・ウォーターフォールと同様な基準でバグを数えると、複数のイテレーションの合計値はウォーターフォールと同様になった。ただし、ユニットごとのエラーをバグとして数えてしまうと、迅速性（アジリティ）を失う。
・アジャイル開発でのバグ傾向の過去のデータがないため、ウォーターフォールの過去の値を参照すると、安心にはなる。
・ウォーターフォールでも、テスト結果は偽造も含め、正しくない場合もなくはない。
・今後、アジャイル開発でのバグ収束曲線に相当するものを定義できるととてもよい。
・ 品質に関する情報は、収集しておき、ユーザーへ提供するとユーザーの理解が得られる。
・テストのためのリソースと時間を掛け合わせると、結果的にウォーターフォールと同じような値になる。

1.5 テストへのインプットについて

・設計書は、テスト、検査、マニュアルへのインプットになる。
・必要最小限の設計書は必要である。
・開発を繰り返して経験を積むと、設計書が少なくなっていく傾向がある。
・ユーザードキュメントについては、複数イテレーション分をまとめて作成した方がよい。

1.6 その他

・ 優秀な技術者であればどのような方法でも開発は成功する。優秀でない技術者が開発をしたときに、ウォーターフォールよりもアジャイルの方が早期に失敗が判明する。

2.人材育成についての検討

2.1 アジャイルプロジェクトのプロジェクトマネージャーについて

・従来型のプロジェクトでのプロジェクトマネージャーの経験がないと、アジャイルプロジェクトを率いることは難しい。
・アジャイルプロジェクトは、チームやペアで仕事を進めるので、仕事の進め方等への改善意識が高まる。
・エンジニアのモチベーションについて
・成功体験が必要である。普通では完了できないスケジュールの中で開発を無事に終了した経験等が重要である。
・プロジェクトマネージャーは、メンバーにどれだけの成功体験を与えられるかがポイントである。ファシリテートや障害を乗り切れる方法等を体験させる。
・アジャイル開発はスキルの低い人をフォローしやすい環境にあるため、スキルアップには役立つ。ただし、理想的なアジャイル開発を体験させる必要がある。
・米国企業のように、スキルを人事考課に反映させる必要があるかもしれない。
・スキルには、大きく分けて、知識とプラクティスの習得の２つがある。特に実際の開発に近いプロジェクトを体験させることにより、プラクティスを習得させることが重要である。
・若いエンジニアがスキルを伸ばすには、成功体験と成功した時にほめることが必要である。以前のように、間違いを指摘するだけではエンジニアを育成することはできない。
・失敗していることを改善するために、プロセス自身を変えていくことにより、エンジニアの開発に対する気持ちが変わっていく。この変化により、スキルが伸びていく。
・アジャイル開発に対応したツールを利用することで、自然にアジャイル開発プロセスになれることができる。教育には、ツールは便利である。ただし、背景となる考え方は教育する必要がある。
・アジャイル開発プロセスを理解する上では、実際に体感しやすい「計画ゲーム」プラクティスを実践する。計画ゲームにより、見積りから計画をたて、実際の実績まで、全体を理解できる。

2.2 開発メンバーについて

・ アジャイルプロジェクトは組織化が行ないやすく、一度経験したメンバーはアジャイル開発を迅速に遂行することができるようになる。
・アジャイルプロジェクトの失敗経験はトラウマとなりやすく、アジャイル開発に消極的になってしまう。ウォーターフォールの場合は、分業化されているので、トラウマになるような致命傷にはならない。
・開発メンバーは、技術力よりもファシリテーション能力が重要である。

・プロジェクトの共同作業により、メンバー個人の能力をすぐに知ることができる。
・開発を繰り返し行なうので、繰り返し（イテレーション）毎に、スキルアップしていくことができる。
・ 開発メンバーの選定基準は、ウォーターフォールとアジャイルでは違いがある。
ウォーターフォールでは、上流担当者はコミュニケーション力、下流担当者はプログラミング力（経歴より）が必要とされ、アジャイルでは両方を要求される。
・技術的スキルが高い人だけでプロジェクトが成功するとは限らない。
・プロジェクト成功のためには、チームとして助け合っていく必要があり、そのような考え方を持つ必要がある。
・ツールの導入だけでは、チームの規模が大きい場合は難しい。プロセスを正確に理解し、チーム間のコミュニケーションをあげることが重要である。
・ウォーターフォール開発から移行しやすいのはスクラムのようなイテレーションが比較的長く、プロジェクト管理に焦点を絞ったプロセスである。

2.3 その他のスキル

・アーキテクトの教育
・アーキテクトはウォーターフォールと同様にプロジェクトに必要である。
・アジャイル開発ではアーキテクチャの原型を早い段階で決めておく必要がある。そのため、ゼロ回目のイテレーションとしてウォームアップやスパイクなどということを行なう。また、はじめのイテレーションには、アーキテクトを投入することも有効である。
・ データベース設計について
・アジャイル開発ではデータベース設計が難しい。特に、既存のデータベースがある場合や性能を要求される場合は難しい。オブジェクト指向開発では、オブジェクトモデルを作成してからデータベース設計を行なうが､日本ではこの方法に慣れていない。
・ また、オブジェクトモデルからデータベーススキーマを生成する優れたツールがあれば、難度は下がるであろう。
・アーキテクトやデータベーススペシャリスト（DBA）は特殊な技量なため、チーム全員に必要なスキルではない。しかし、チームには欠かせないスキルである。

2.4 作業の粒度

アジャイル開発では、作業を１日で終わるような単位に分割する。作業分割スキルを身につける方法はあるか。

・フレームワークの活用により、作業を細かくできるようにする。
・テスト駆動開発の導入により、作業分割する。
・作業分割（タスク分割）は、実際に体験してみて、スキルとして身につけていく。
・ペアプログラミングを導入して、身につけていく。
・タスク分割はチームとして行なうものとして、タスクは上限４時間までの大きさにするなどのルールを決めても良い。
・プロジェクトマネージャーが予測と実際との比較を行う。同じになっている場合は、何かがおかしいことが多い。
・作業（タスク）の消化状況は、張り出す等して、見える化する。
・ 作業分割方法の事例等、過去の経験を共有することも重要である。

2.5 日本におけるアジャイルエンジニアの育成

・ 日本でアジャイルエンジニアが自然に増えるとは思えないので、教育の仕掛けを作る必要がある。能動的なチャレンジ精神をどのように養うかを考える。

・日本では製品パッケージの開発が少ないため、アジャイルエンジニアの育成に大きな投資ができないことが多い。
・大規模システム開発の成功だけではなく、新規ビジネス開発等にも報奨金を出す等のしかけが必要である。
・欧米とは違った日本的なアジャイルはあるはずである。品質、テスト、データベース設計、設計書の標準化など、日本のソフトウェア開発がすぐれているものに注目する。
・コンシューマー向けのシステムはアジャイルプロセスで開発されているはずなので、アジャイル開発の経験値はある。
・日本のソフトウェア業界が、市場を広げるために、アジャイル開発を推進していくかどうかが重要である。
・日本の場合、クラウド環境になっても、アジャイル開発の必要性を認識しないかもしれない。それは、海外との競争がないシステムが多いからである。ただし、開発費のコストダウンは迫られるであろう。上流だけ日本で行なって、開発は海外にシフトされ、日本から開発がなくなることもあるかもしれない。
・SalesForce.com のように、常に開発していくシステム（フレームワーク）はアジャイル開発が最適なので、このような形態が多くなるとアジャイルも多く用いられるかもしれない。
・日本の場合は、縛りを入れて管理するアジャイルプロセスがよいかもしれない。すなわち、日本的なアジャイル開発プロセスが必要になる。

3. 契約についての検討

3.1 契約形態（構内請負）について

・契約形態としては、外部受託、構内請負、派遣契約等がある。
・外部受託、派遣契約は契約的にはアジャイル開発を行うことが可能である。
・構内請負の契約では、常駐責任者を通してでないと指示ができない。
そのため、アジャイル開発で最も重要なコミュニケーションができなくなる。
・構内請負と支援契約を組み合わせる方法もある。

3.2 受注者、発注者側のリスク

・受注者としては、期間内に開発するシステムの要件を決めたい。
・リスクヘッジとして、手続きは従来通りにすることも考えられる。
・受注者としては、営業的に請負の方が、利益が大きい場合もあるので、準委任にしたがらない。支援契約の単価設定は難しい。コンサルタントとして認められるのは難しい。

3.3 その他

・発注者側がアジャイル開発を希望したときには、受注者側が、営業、契約、プロセスなどの提案作業ができていればビジネスチャンスになる。
・現在の開発では、多くの機能を開発しすぎる傾向にあるが、アジャイルプロセスで必要な機能だけを開発すれば、適正なシステムが完成する。
・ウォーターフォール以外の契約的な側面は、民間企業だけではなく、国が先導して整理すべきである。
・外部受託でアジャイル開発をする場合は、週１回の電話会議等により、きっちりとした調整が必要である。
・１イテレーションを請負として、細かく契約することも考えられる。また、リスク単位での契約も考えられる。

・日本では、日本的な利害調整、コミュニケーション、エビデンスがあり、米国のようなアジャイル開発は難しい。
・米国の場合、成功報酬のような制度があるが、日本は減点主義なので、アジャイル開発成功時の開発費用の増額は難しい。
・ウォーターフォールとアジャイルの中間的な形態をとりながら、段階的にアジャイル本来の契約形態に近づける方法もある。
・アジャイル開発の知識があり、開発能力が高くないと、トラブルになるかもしれない。
・請負の場合、要件を決めてから開発を行っていくため、アジャイル開発には向いてないと考えられている。しかし、実際には、書籍等で書かれているようなストーリーから開発を始めるわけではなく、従来開発の大項目に相当するものは決めるので、この考えは間違えである。要件のまとめ方は、各社のノウハウであろう。
・リスクを考慮すると、アジャイル開発といえども、はじめの見積りは高めに設定しなければいけないかもしれない。
・経済危機により、ユーザー企業の内製化が進んでいる。アジャイル開発の特徴であるユーザー主体の要件出しがフィットするかもしれない。
・大規模なプロジェクトよりは、小規模なプロジェクトが増えているので、アジャイル開発には向いている開発が増えている。また、アジャイル開発を受け入れる土壌ができだしている。
・アジャイル開発に対する知識が二極化している。日本では、ウォーターフォールの要素を取り入れたアジャイル開発が向いているので、極端な考え方は訂正すべきである。