デジタルプラクティス　Vol.11 No.2(Apr. 020)

特集号招待論文

モールを用いたプログラミングによるアジャイルマインドの学習プログラム ―体験を通したアジャイル開発の実践的理解―

田中 貴子[1]　斎藤 忍[2]

[1]NTTテクノクロス　[2]NTT

プロダクト開発のアプローチとしてアジャイル開発のニーズが高まっている．ウォータフォール開発とアジャイル開発は採用するプロセスやツールの違いもあるが，プロジェクトのメンバが理解し，実践すべきマインドセット（重視すべき考え方）が大きく異なっている．開発者・ユーザは，アジャイル開発のマインドセットを実践する上での留意点（起こり得る苦労）を正しく理解した上で，アジャイル開発の導入を進めることが必要である．本稿では，アジャイルマインドを短期間で習得するために筆者らが独自に開発した学習プログラムの実施内容と評価結果を報告する．学習プログラムでは，モールを活用した開発の疑似体験と振り返りを行い，アジャイル開発への過度な期待感を是正することを狙いとしている．

1. はじめに

近年，市場や環境の変化が加速し，ディスラプタによる業界参入の脅威などにより，企業はIoT（Internet-of Things）やデジタルトランスフォーメーション（DX）などへの迅速な対応が求められている．一方，プロダクト開発のアプローチとしてのアジャイル開発はIoTやDXとの親和性も高く，適用のニーズが高まっている．

これまでの国内企業における開発のアプローチでは，ウォータフォール開発が数多く適用されてきた．ウォータフォール開発とアジャイル開発のアプローチでは採用するプロセスやツールの違いもあるが，プロジェクトのメンバが理解し，実践すべきマインドセット（重視すべき考え方）が異なっていることも挙げられる．

ウォータフォール開発の基本的な前提は，要求が実装前にすべて決定し，詳細化されていることである．そのため，要求をすべて（網羅的に）プロダクトとして完成した上で，（納期までに）遅れずにユーザに届けることが重視される．一方，アジャイル開発ではユーザ自身が必ずしも本当に欲しいものを精緻に定義できていないことを前提としている．アジャイル開発では，最小限の要求をプロダクトに作り上げた上でユーザに早く届け，そこから得られるフィードバックに基づきプロダクトをより良くしていくことを重視する．

アジャイル開発のマインドセット（以降，アジャイルマインド）を言語化したものに「アジャイルソフトウェア開発宣言」[1]がある．この宣言では，アジャイル開発で価値をおくべきことがらとして，「プロセスやツールよりも個人と対話を，包括的なドキュメントよりも動くソフトウェアを，契約交渉よりも顧客との協調を，計画に従うことよりも変化への対応を，価値とする」と述べている．

上述のアジャイルマインドは平易な文章であるため，開発者やユーザにとって受け入れやすい内容である．一方，筆者らの意見として，アジャイルマインドの内容は受け入れやすい反面，アジャイル開発に対して過剰な期待感（例：動くものがすぐできて，自由にものも変えることができる）などを抱いてしまう可能性もある．当然ながら，ソフトウェア開発プロジェクトで採用するプロセスやツールを，アジャイル開発のアプローチに変更するだけで，アジャイルマインドで示された価値を具現化することは難しい．開発者・ユーザは，アジャイルマインドを実践する際の留意点を正しく理解した上で，アジャイル開発を進めることが必要である．

そこで本稿では，アジャイルマインドを短期間で習得するために筆者らが独自に開発した学習プログラムの手法と201名の受講者（22回の研修を実施）から得られたデータよりその効果を報告する．本稿の構成は以下のとおりである．第2章では，アジャイルマインドを実践する際の課題を述べる．第3章では，アジャイルマインドの学習プログラムの特徴を述べる．第4章で，4時間の学習プログラムの構成を紹介する．第5章では学習プログラムの効果を考察し，最後の第6章でまとめを述べる．

2. アジャイルマインド実践の落とし穴

実際のソフトウェア開発では，当初想定していた前提（例：想定ユーザ，利用シーン）が維持された状態で開発が進むことはきわめて少なく，外部環境の変化等により前提の内容が新しく追加されたり，変更されたりするケースが一般的である．これにより，開発期間の中で要求の変更が発生する．プロジェクトはこれらの要求の変更に対して，何らかの対策を実施する必要がある．主な対策は，（1）要求の変更を（一部も含めて）受け入れて作り上げる，（2）プロジェクトの開発期間中には要求の変更を受け入れず，次期以降に先送りするなどの調整を行うことである．（2）の場合プロジェクトが採用しているアプローチがウォータフォール開発であるかアジャイル開発であるかの如何にかかわらず，要求の変更発生時には，実施すべき対策は意思決定する必要がある．

筆者らは，前章で述べたアジャイル開発のアプローチに対する過剰な期待感が，プロジェクトの意思決定に悪影響を及ぼす局面が少なからず存在していると考える．アジャイル開発を導入したプロジェクトリーダ・推進者との意見交換を通してよく聞かれた代表的な意見を**表1**に示す．アジャイル開発を導入した際に苦労した点について，3つの観点（要求・プロセス・役割）に分類することができた．

表1　アジャイル開発の初期導入時に起きがちな苦労

要求の観点
● 対応が難しい無理なタイミングで無理な要求変更をしてしまう
プロセスの観点
● ウォータフォール開発のようにドキュメントの品質に力を入れ過ぎてしまう ● 計画駆動型の習慣から，顧客のフィードバックループを活かさず，当初計画のものを作ろうとする ● 「プロダクトバックログ」で書かれているものを全て作ろうとしてしまう
役割の観点
● スクラムマスターが，プロジェクトリーダのようにふるまい，チームの作業を割り振って指示する ● プロダクトオーナーが君臨し，開発メンバーとの相談をせず，決定事項として要求だけを突きつける

表1の「要求の観点」で記している苦労（対応が難しい無理なタイミングで無理な要求変更をしてしまう）が，前章で述べた，ユーザのアジャイル開発に対する過剰な期待感に起因したものと考えられる．過剰な期待感を持ったユーザは，アジャイル開発がどの時期でもどのような要求変更でも可能であると捉えてしまう．そのため，本来は要求変更が不可能なタイミング（例：開発の後期）で，プロジェクトは技術的や工数的にきわめて実現困難な要求変更に対応することになる．結果として，プロジェクトの失敗につながる．このように，現実的に対応が難しい要求変更を受け入れてしまえば，たとえアジャイル開発であっても実際のプロジェクトは失敗する．アジャイルマインドを実践する上では，実際に起こり得る苦労を事前に十分に理解しておく必要がある．

表1のプロセスや役割の観点も実際に起こり得る苦労ではある．一方，前述の意見交換を通じて，筆者らは，アジャイル開発への過度な期待感を是正するには，まずは要求の観点で述べた苦労を体感・学習することが重要であるとの考えに至った．このような考えのもと，要求変更への対応を体験できる学習プログラムの開発を行った．次章以降で学習プログラムの内容を記していく．

3. 学習プログラムの特徴

アジャイルマインドを学習する本プログラムは，アジャイル開発の過度な期待など誤解に気付くための仕組みを取り入れた学習プログラムである．座学ではなく，受講者自身が体験を通してアジャイルマインドを自ら学ぶ．

学習プログラムの目標は，変化への対応を通して，アジャイルマインドの理解と，受講後に，アジャイル開発について，自分の言葉で第三者に話すことができることを設定している．それらの達成目標を受講者ができるように，学習プログラムにはいくつかの仕組み（特徴）を組み込んでいる．

3.1　特徴①：2つの開発プロセスの同日体験と振り返り

アジャイルマインドの学習プログラムは，アジャイル開発とウォータフォール開発の2つのプロダクト開発を同日に実施する．開発のアプローチによる変化への対応の違いについて，体験を通して理解するためである．さらに，開発手法の違いについて体験したことを直後に言語化して振り返ることを繰り返す．自分の体験を言語化して自分へのフィードバックをすることと，他の受講者と共有することによって，受講者に対して体験を通した気付きと知識の定着を図る（**図1**）．

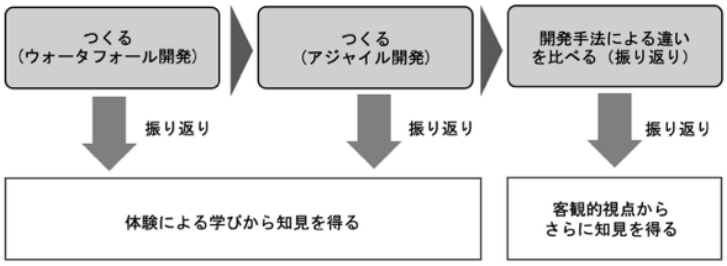

図1　受講者の学習体験

3.2　特徴②：仮想ユーザに向けたプロダクト

アジャイルマインドの学習プログラムでは，運営側が仮想ユーザを設定し，仮想ユーザの要求を受講者に提供する．その要求に応えるプロダクトを受講者がチームでつくることになる．こうすることで，チームはユーザに対してプロダクトを開発するという状況を再現している．

受講者はチーム内で2つの開発プロセスごとに担う役割がある．ウォータフォール開発では発注役と開発メンバ役，アジャイル開発では，プロダクトオーナー役と開発メンバ役に分かれる．

仮想ユーザの要求は「乗り物だけではなく売店もある遊園地の風景がほしい」というような形で発注役とプロダクトオーナー役の受講者に提示される．発注役とプロダクトオーナー役が，常に仮想ユーザの要求を意識してプロダクトをつくる状況をつくり，自分の裁量や好みでプロダクトのゴールを設定しないようにしている．

3.3　特徴③：プログラミングを疑似的に再現

受講者の意識が開発のプロセスに向くように，プログラミング言語を書かず，手芸用モール（以下，モールと略す）を用いたプロダクトの開発を採用した（**図2**）．モールは，長さ30cm程度でカラー合成繊維の中心に針金が通っている，折り曲げ可能な子供用玩具である．

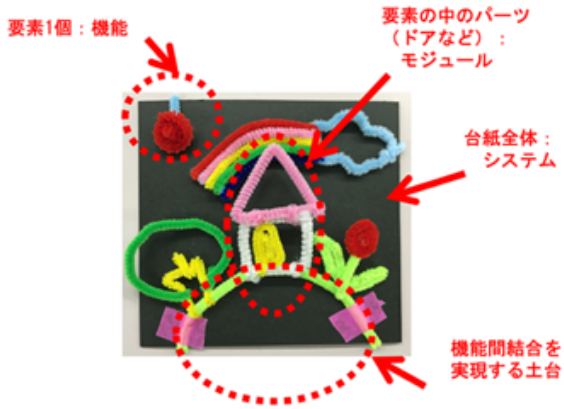

図2　手芸用モールによるプログラミング要素の再現

実際のプログラミング言語でソースコードを書くと，ソースコードを綺麗に書くことや，プログラムのコンパイルを成功させることの方に意識が向きやすい．一方，モールはきわめて安価に入手可能であることに加え，プログラミングの条件（**表2**）を以下のように満たすことができる．

表2　プログラミング再現の条件

観点	状態
プログラミング言語の再現	・開発するプログラミング言語は知っているが，これから作り上げるものはまだわからない状態が再現できる ・プログラミングの能力差のような個人差が出る
プログラミングするプロダクトに対する再現	・全体(システム)があり，それらをいくつかの構成要素(パーツ)に分割できる ・分割した要素(パーツ)は，最終的に結合することで，全体を構成でき ・不具合が生じた場合，修正・是正ができる

- 特にレクチャーをせずとも，モールをねじる，切るなどによって何等かの形状を容易に作ることができる．さらに，手先の器用さにより，プログラミング能力のように生産性や品質の個人差が生じる
- パーツごとにつくってから，それらを最終的に結合できる．さらにそれを修正可能である

受講者は，プロダクトの製作中は，材料（モール）および道具（ペンチ）に集中しすぎることなく，プロセスに注力できる．参加する上での制約がプログラミング能力に依存しないことから，開発者以外の営業担当，企画担当，マネージャなども参加して学ぶことができるようになる．

3.4　特徴④：揺らぎ（仕様変更）の発生

本学習プログラムでは，受講者に仕様変更の対応を求めることで，受講者のチームに揺らぎを与えている．

チームの進捗状況を加味して，仮想ユーザの要求が変わることによる仕様変更対応を各チームに求める．仮想ユーザの要求の変動パターンは，ソフトウェア開発のよくあるケースをモデル化したものを数種類用意している．具体例を**表3**に示す．

表3　実際に使用している仮想ユーザの要求例

モデル化した仕様変更パターン	
基本コンセプトは変更がないが，外部環境の変化などによって，コンセプト要素の追加や機能追加が発生	
本学習プログラムで受講者に提示する仮想ユーザの要求	
【当初】	乗り物だけではなく売店もある遊園地の風景がほしい
【仕様変更後】	やっぱり遊園地の風景でも，たくさんのアトラクションある楽しそうな遊園地での家族の記念写真のようにしたい

受講者は，表3のようなユーザ要求の変化にチームで対応していく．本学習プログラムは，1回あたり複数のチーム構成で行い，最後に全員比較することをあらかじめ受講者に伝えてある．そのため，その変化に対応する必要性に迫られる中で，チームに競争意識も生まれる．それによって，受講者は自然に学習プログラムに没入し，主体的に取り組む形になり，体験を通した知識がより記憶に強く残る．

4. 学習プログラムの構成

ウォータフォールプロセスでの開発，アジャイルプロセスでの開発をそれぞれ体験し，それを開発手法による違いを振り返るという構成で，全体で4時間である．それぞれの開発プロセスの体験は，モールによる開発を行う「つくる」と自分たちの活動を「振り返る」の2部構成である．1回の研修は2チーム以上で実施する．アジャイル開発のプロセスとして適用率の高い[2]スクラムを採用している（**図3**）．

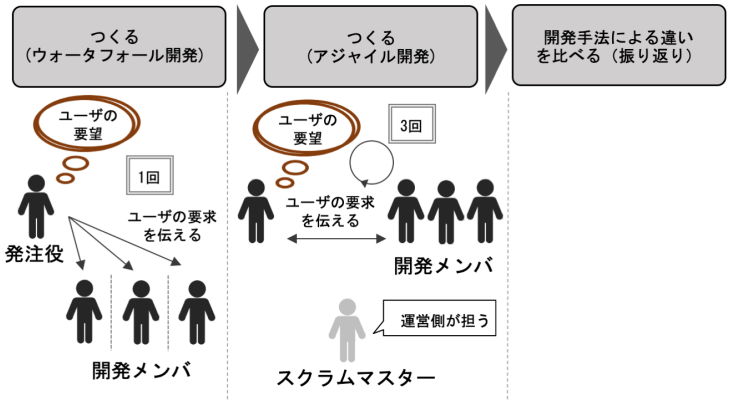

図3　プログラムの構成

どちらの開発も，はじめに運営側が，発注役とプロダクトオーナー役に仮想ユーザの要求を提示する．その要求を踏まえて，発注役やプロダクトオーナー役がつくるプロダクトを考え，開発メンバ役の受講者に伝える．開発メンバ役は，プロダクトの情報を元にモールを用いて要求を出した仮想ユーザに届けるプロダクトを製作していく．

4.1　つくる（ウォータフォール開発）

ウォータフォール開発は，一般に，発注者と開発担当の階層構造を持ち，発注者から提示された要件に従って，期日までに担当機能ごとに開発していく．本プログラムでは，その特徴を模擬して強調するために，発注役は開発要件を1回だけ開発メンバに伝え，開発メンバは，それぞれ独立した開発担当とし，サイロ化した形で独立に開発する設定とする（**図4**）．受講者に，それらの役割をあらかじめ決定してもらう．

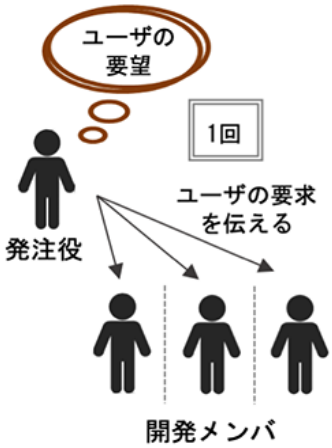

図4　つくる（ウォータフォール開発）における役割

モールは作った形状がすぐに判断できるため，ウォータフォール開発のプロセスを簡略化し，最後に結合させる部分をテストに集約して行う（**図5**）．

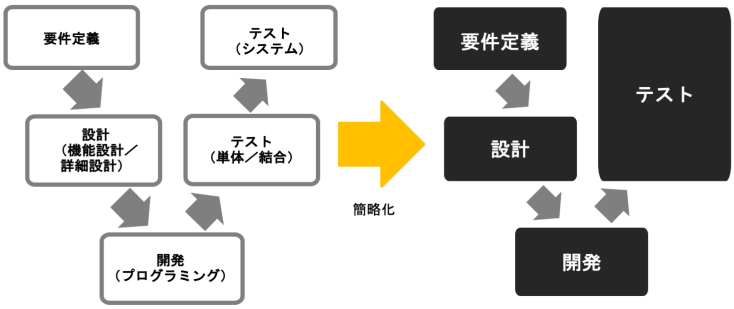

図5　開発プロセスの簡略化（ウォータフォール開発）

4.2　つくる（アジャイル開発）

アジャイル開発は，階層構造を持たず，フラットなチームで密にコミュニケーションを取りながら協調開発を行う．その特徴を模擬するために，開発メンバが対等な関係のコミュニケーションとなるよう運営側がファシリテーションをする．また，開発メンバ間のコミュニケーションを活発にするため，それぞれが独立に開発を行うのではなく，同一空間で協調しながら開発する設定とする（**図6**）．受講者には，これらを行うプロダクトオーナー役と開発メンバ役をあらかじめ決定してもらう．同日内にスプリントを3回行うため，アジャイル開発は，デイリースクラムを省略し，それ以外のセレモニーはすべて行う（**図7**）．

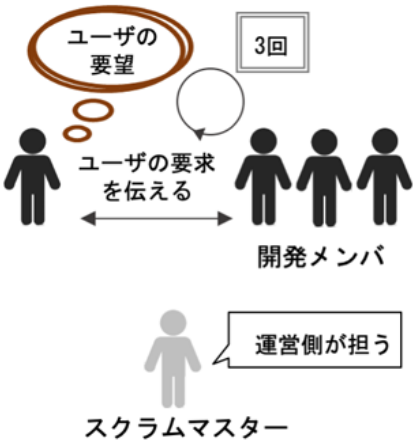

図6　つくる（アジャイル開発）における役割

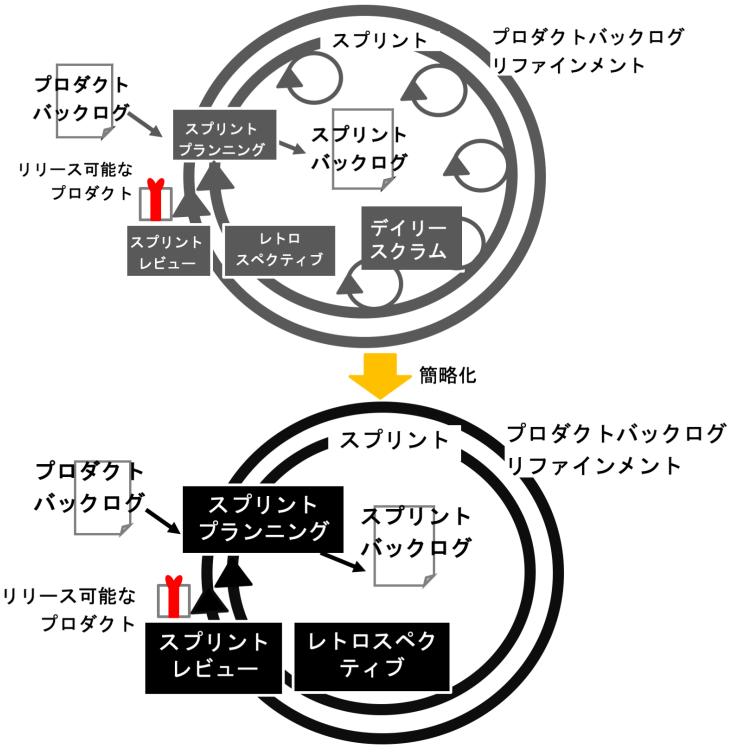

図7　開発プロセスの簡略化（アジャイル開発）

1回あたり複数チームの構成で実施するため，モールによるプロダクトが毎回複数グループできる．それらを受講者全員で見比べていく．

4.3　振り返る（開発アプローチによる違いを比べる）

本学習プログラムは，１回あたり複数チームでプロダクト開発を行うため，ウォータフォール開発とアジャイル開発ごとに，チーム分のプロダクトができあがる．受講者全員で仮想ユーザの要求を共有した上で，できあがったプロダクトを比較する．比較観点を**表4**に示す．これらの観点を受講者全員で対話をしながら比較し，体験や気づきを言語化して，共有していく．

表4　プロダクトを比較する観点

● 揺らぎの対応を直感的な印象から比較する ● 直感的な違いの理由を考える

実際に仮想ユーザの変化する要求（表3）に対応しながら受講者が制作したプロダクトを，**図8**，**9**，**10**，**11**に示す．

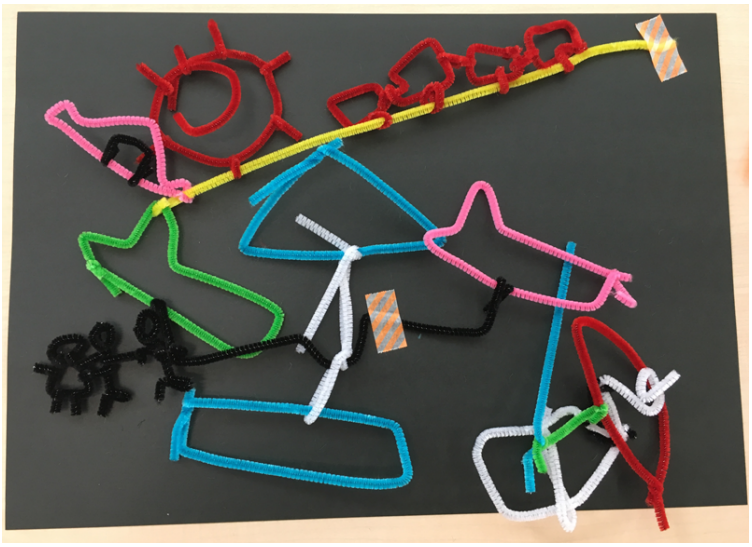

図8　ウォータフォール開発による制作

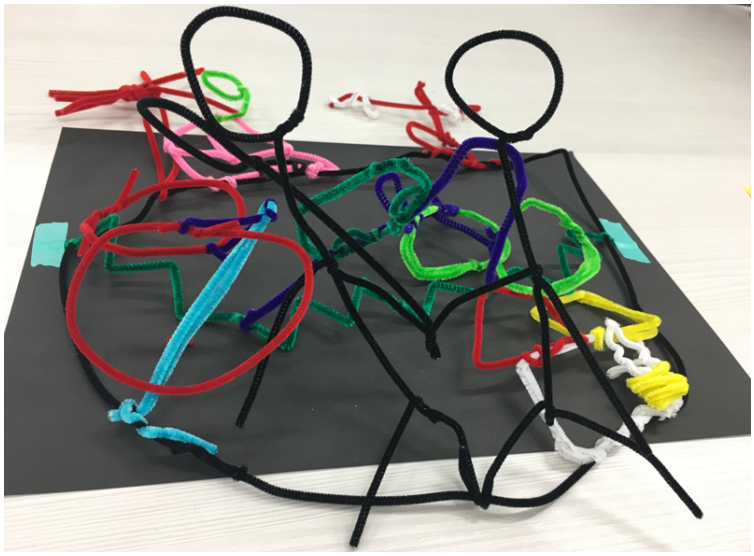

図9　ウォータフォール開発による制作

図10　アジャイル開発による制作

図11　アジャイル開発による制作

5. 学習プログラムの効果

筆者らが第4章で述べた学習プログラムにより開催した22回，201名のアンケート結果から考察する．本プログラムの受講者（アジャイル開発未経験者（座学等による学習経験があるが，実務経験がない場合を含む）とアジャイル開発実践者）のアンケート結果を以下に示す．

5.1　プログラムの評価

5.1.1　本学習プログラムの楽しさ

筆者らは取り組む作業に対して関心や意欲が高ければ，学習効果が高くなることが期待できると考えた．そこで，本学習プログラムでの受講者に対して，どのような意欲で取り組むことができたかを調査した（**図12**）．

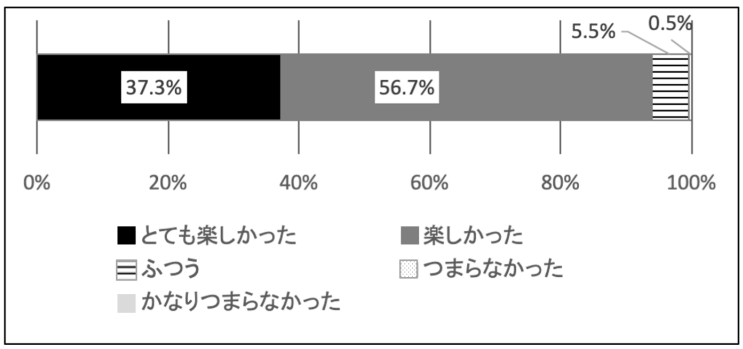

図12　楽しく学べたか（n=201）

受講者が楽しく学べたかについて5件法（とても楽しかった，楽しかった，ふつう，つまらなかった，とてもつまらなかった）で評価させたところ，94%の受講者が「とても楽しかった」もしくは「楽しかった」を選択しており，本学習プログラムに対して，受講者が，関心と意欲を高く持って参加しているといえる．

自由回答でも下記のような回答を受講者から得ている．

- スクラムの楽しさが分かった気がする
- プログラミング開発のアナロジーとしてモールを使うのは斬新で興味深い．「あるある感」を強く感じて楽しかった
- 実際に手を動かすこと，成果が目に見えることで，“アジャイルとは何か”を体感できたように思う

5.1.2　アジャイル開発に対する受講前後のイメージ

筆者らは，受講者のアジャイル開発に対する受講前後のイメージの違いについて調査をした．

本学習プログラムでの受講者の63.7%が，あらかじめ理解していたアジャイル開発が過度の期待や，誤った認識であったと体験を通して自ら理解している（**図13**）．

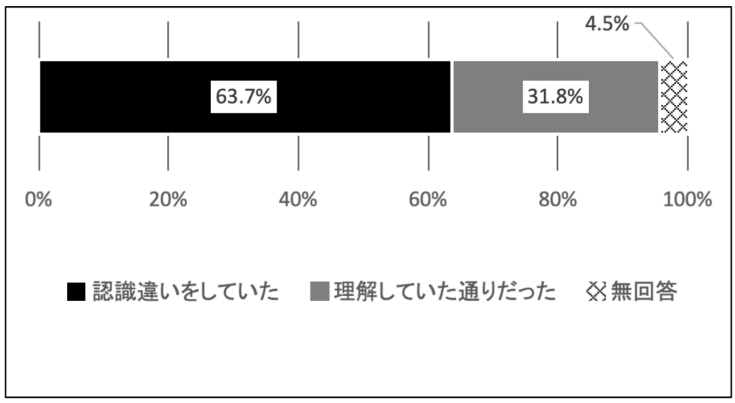

図13　アジャイル開発の理解（n=201）

5.1.3　実践に向けた実業務への活用知識の獲得

本プログラムはソフトウェア開発を再現していることから，得た体験や知識が，受講者の通常業務で活用できるか否かについても調査した．これは，実業務で実践するきっかけを受講者が得ていたかを確認するためである．

実務への活用有無を選択回答してもらった．本プログラムの受講者の86.6%が実務への活用できると回答している（**図14**）．このことから，受講者が本プログラムで実務へフィードバックできるヒントを得ていると考えられる．活用できると回答した理由については，自由回答で記入してもらった．自由回答については次の5.2節で考察を行う．

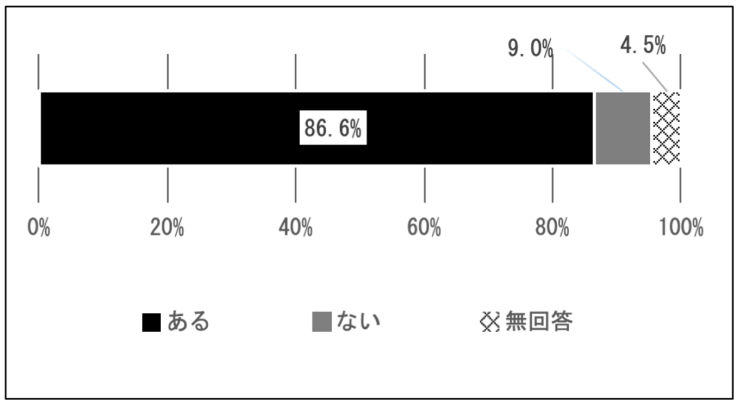

図14　実業務へのフィードバック有無　（n=201）

5.2　受講者が得た知見

アジャイル開発に関してどのような知見を得たのか，アンケートの自由記入欄から考察する．受講者は，単にアジャイル開発やスクラムを理解するだけでなく，より具体的に理解できていると考えられる．

5.2.1　アジャイル開発を適用するだけではうまくいかない

第2章で述べたように筆者らは，アジャイル開発をすることで何でも柔軟に変更できるなど過度の期待がプロジェクトの意思決定に悪影響を及ぼす局面が少なからず存在していると考え，本学習プログラムを開発した．受講者の自由回答から，アジャイル開発の適用可能範囲など深い理解を受講者自身が得ていると考える．代表的な回答を**表5**に記載する．

表5　アジャイル開発適用に関するアンケート回答

どんな変更にも対応できる訳ではない
● アジャイル開発は，変更に柔軟ではあったが，すべての変更に対応できるものではない点，ただし，品質を担保する仕組みはきちんと存在する
アジャイル開発に向き不向きがある
● アジャイルもウォータフォールもメリット，デメリットがある ● アジャイルも向き不向きがある点 ● 個人及びチームの特徴で向き不向きがある ● チームやメンバーのスキルにより，アウトプットが大きく左右されるので，見極めも重要
単なるアジャイル開発の採用だけではうまくいかない
● スクラムの進め方次第では不出来になる ● 本質的に，アジャイル開発も，ウォータフォール開発も万能ではなく，実装者のスキルや発注役やプロダクトオーナーのスキルに依存するということは同じだった ● アジャイルは，適宜見直しができる(イメージと開発内容のすり合わせ，開発作業者の能力振り返り)が，見積もりを事前に固める必要なことや，ステークホルダーが多いと導入が難しいと感じた ● あたらしいことをやる＝アジャイルではない．チームの作り方がアジャイルだとより重要だ

5.2.2　アジャイル開発実践者と未経験者の気づき

アジャイル開発の実践者と未経験者が，本学習プログラムで同じ場で受講した場合でも，受講者，自分自身で理解を得ている．アンケート回答のいくつかを**表6**に示す．

表6　アジャイル開発実践者と未経験者の気づきに関するアンケート回答

アジャイル開発実践者の気づき
● 業務ではプロダクトオーナーとなっており，改めてその役割を認識できた．サービス価値を高める，優先付け，時間と実現機能のトレードオフなどが体感できた ● 今後新しく開発する案件があった際に，ウォータフォールとアジャイルの特性に基づき，どちらの開発スタイルを取るべきか，案件に応じて採用していきたい ● ドキュメントなどコーディング以外の場でもアジャイルな進め方を取り込んでみる
アジャイル開発未経験者の気づき
● 発注者が求めているイメージと，ベンダーが開発しようとしているものとの不一致が起こりえること．アジャイル開発を経験したことは無いが，ウォータフォール開発においても互いの認識がずれないよう注意を図るべき ● 最初のコミュニケーションで機械的に要件を伝えるだけでなく，目的や完成イメージに対する「想い」を伝えることで開発者が持っている未知の技術や様々な実現方法の提案を引き出せる ● とにかくコミュニケーションが大切であるということ．"すり合わせ"の大切さ ● チームビルディングはとても良いと思う．特に密なコミュニケーションの必要性を理解してもらうのに良い．ウォータフォール開発でも使える

また，開発担当以外も受講があった．共通言語としてのアジャイル開発の考え方の共有にも効果があると確認できた（**表7**）．

表7　開発業務以外の受講者の気づきとアジャイルの別分野への適用

開発業務以外受講者の気づきとアジャイルの別分野への適用
● 知識がなかったが，イメージがわいた．自分は営業業務であるが開発との議論で，目線合わせしやすくなった ● アジャイルの考え方は開発だけでなく，制度(運用ルール)設計にも取り入れられないか ● 営業担当として，サービスの提案をしている．「クライアントの要望＝すべて網羅しなくてはならない」ではなく，クライアントに「さほど重要でない要素は限られたリソースの場合」一旦あきらめてもらう」方が，運用で苦しまないのでは，と確信した

5.3　アジャイルマインド習得の汎用性

アジャイル開発のプロセス，プラクティスは比較的導入しやすいが，マインドセットは習慣化した思考に左右されやすく，すぐには変わらない．アジャイル開発のマインドセット（アジャイルマインド）の理解をする場は実際のソフトウェア開発プロジェクトがほとんどであり，実践できるようになるためには，時間と経験が必要である．本稿で紹介した学習プログラムは，プロダクト開発の疑似体験を通して学ぶことを可能とする．

アジャイル開発というと開発者の世界に限定されるイメージがある．しかし，アジャイル開発をしている開発者の周囲がウォータフォール開発のマインドセットで動いていると，開発者だけがアジャイル開発のマインドセットになったとしても，アジャイル開発の実践は困難である．開発者の周囲にいる，たとえば，営業担当，企画担当，マネージャ等も，アジャイル開発の理解者となっている必要がある．本稿で紹介した学習プログラムは，モールによる疑似的な開発体験のため，これら開発者の周辺の方々でも受講可能である．

6. おわりに

本稿ではプログラミングをモールで疑似的に再現したプロダクト開発の体験から，ウォータフォール開発の知識と結び付けて誤解するリスクを下げ，アジャイル開発のプロセスとその背後にある原理・原則などを体感して理解できる手法を紹介し，201人の受講者から得られた効果から，本学習プログラムがアジャイル開発のマインドセットを理解する上で高い効果があることを示した．

アジャイル開発のプロセスやウォータフォール開発は，実はソフトウェア開発に限ったものではない．企画し，設計し，制作するというプロセスは，モノづくり，コトづくりのプロセスでもあるため，アジャイル開発を知るということは，ソフトウェア開発者以外にも有益である．本プログラムはソフトウェア開発者だけではなく，プログラミング能力に関係なく参加できるようにしている．

今後の予定として，本プログラムを，アジャイル開発の理解を核としたさまざまな分野に拡張することを検討している．その第1弾として，アジャイル開発のマネージャ層向けにカスタマイズする方法を試行中である．また，デザイン思考の研修と組み合わせることで，ユーザニーズの理解からサービスの具体化までを一気通貫で学べる学習プログラムの構築にも取り組んでいる．いち早くユーザにサービスを届けてフィードバックから良いプロダクトを作っていくアジャイル開発が，ウォータフォール開発のように当たり前となる世界になるように貢献していきたい．

参考文献

1）アジャイルソフトウェア開発宣言：http://agilemanifesto.org/iso/ja/manifesto.html (2019年11月15日現在)
2）13th Annual State of Agile Report：Vision One (2019).

田中 貴子（非会員）tanaka.takako@ntt-tx.co.jp
NTTテクノクロス（株）こころを動かすICTデザイン室に所属．Scrum Alliance認定スクラムマスター．大小さまざまなソフトウェア開発，アカウント営業，イントレプレナーを経て，ビジネスとDevOpsの一体化をテーマに研究するに至る．現在は，こころを動かすICTデザイン室において，これまでの知見とUX，デザイン思考，アジャイル開発などを駆使したこころを動かすサービスデザインに従事．

斎藤 忍（正会員）shinobu.saitou.cm@hco.ntt.co.jp
日本電信電話（株）ソフトウェアイノベーションセンタに所属．ソフトウェア工学の研究開発に従事．博士（工学）．

採録決定：2020年2月7日
編集担当：飯村 結香子（NTT ソフトウェアイノベーションセンタ）