ITC大阪城

デザイン思考等の考察
報告書

2022年１月
ITC大阪城

はじめに

新型コロナウィルスにより、ビジネス環境は大きく変化している。AI、IoT や RPA 等の言葉に引きずられるのではなく、原点に立ち戻り、IT 経営を見つめ直す事が大切と考える。

このための一つのアプローチとして、IT コーディネータとしてデザイン思考を考えることは IT 経営推進に有効な手法である。また、プレゼンテーションやワークショップについても新たな知見をもつことが必要と考えている。

今回のテーマ研究調査活動は、デザイン思考をナレッジ、だますプレゼンをハウツウ、モブプログラミングをメソッドとしてアプローチしている。

(1) Knowledge
デザイナーやクリエイターが業務で使う思考プロセスを活用し、前例のない課題や未知の問題に対して最適な解決を図る思考法であるデザイン思考をナレッジベースとして知見することを目的とする。

(2) How To
プレゼンテーションが正しく伝わるか、誤った理解をされないかを逆説的にだますプレゼンテーションの方法から知見する。

(3) Method
システム開発のアジャイル開発においてモブプログラミングがある。これを自らのワークショップ活動に適用するために、モブプログラミングについて知見する。

なお、今回の報告書は、各編独立したものを合本した形式となっている。個々の担当者の特長が表れていると考えている。

本報告書がITコーディネータの知見の一助となれば幸いである。

最後に、本テーマをテーマ研究調査活動としてご承認くだされた特定非営利活動法人ITコーディネータ協会に感謝いたします。

2022年1月 18 日

ITC大阪城　テーマ研究調査活動WG
リーダー　　新保　康夫

目次

第Ⅰ編

デザイン思考とは／創造的解決策を導くアプローチ

山中美智子

第Ⅰ編　デザイン思考とは／創造的解決策を導くアプローチ

目次

1　デザイン思考とは

1.1　デザイン思考という言葉

デザイン思考とはどのような思考であり、どのようなアプローチを行うのだろうか。まず、言葉の意味から考えてみる。

「デザイン」は、国語辞典によると(1)「設計」(2)「図案、意匠」とある。ともすればグラフィック描画のことだけを示すイメージで捉えられることがあるが、色や形、配置なども含めて「設計」である。デザインをする人のことを「デザイナー」と呼び、服飾・建築分野のみならず Web や IT 業界等で設計を行う職種を指す。

「デザイン思考」とはデザイナーのように設計をしながらプロセスを取り入れ、設計をしながら問題解決していく思考プロセスである。

1.2　デザイン思考のアプローチ

デザイン思考のアプローチは、次のように定義される。

- 人々が生活のことで何を欲し、何を必要とするか。
- 製造、包装、マーケティング、販売、および、アフターサービスの方法について、人々が何を好み、何を嫌うのか。

について、直接観察し、徹底的に理解し、それによってイノベーションに活力を与えること。

このことをエジソンによる電球の発明を例にして、もう少しデザイン思考とは何かを紐解いていこう。

エジソンは、まず電球を発明するというイノベーションを起こした。しかし、ただ電球だけがあっても、電球を使用する社会がなければ無用の長物となってしまう。そこでエジソンは、電球を使用する社会の実現のために電力システムを創出した。

エジソンは、発明品のみならず、完全に発達した市場をも思い描く想像力を持っていた。すなわち、人々が自分の発明品をどのように使いたいと思うのかを想像できた為、

電力システムのようなものを実現できたのである。彼はユーザーのニーズや嗜好を必ず検討し、イノベーション活動の全領域に人間中心のデザインの真髄を吹き込むアプローチをとっていた。これがまさしくデザイン思考のアプローチの、初期の例と言える。

エジソンのアプローチは、事前に考えられた仮説を検証することを目的にするのではなく、試行錯誤による挑戦から、実験家たちが新しいことを学べるように支援することを目的にしていた。際限なく繰り返される試行錯誤は、まさに「99%の努力」だったのだ。

このように、イノベーションプロセスが多大な試行錯誤とデザイナーの感性と手法を用いてユーザニーズや嗜好に沿った文化と技術の力を取り持つことで、現実的な事業戦略や顧客価値・市場機会を創出する。

以降の節では、様々な問題解決アプローチの事例を紹介しながら、デザイン思考について考察していく。

1.3　デザイン思考の 3 段階のプロセス

偉大な発明・アイディアは、ともすると天才の神業的な想像力で作られた完璧なものであると捉えられていることが多い。しかし、実際には神業でも天才のひらめきでもなく、人間中心の発見プロセスによって創造的にプロトタイピング・検証・改善といったサイクルを何度も繰り返したことで完成していくのである。

デザイン思考のプロセスは体系的なステップを進むのではなく、複数の「スペース」で構成される。各スペースを通過し、あるいは、行ったり来たりしながら進めていく。スペースとは、次の 3 つである。

(1) 着想（インスピレーション）：解決策のあくなき探求を動機付ける。
(2) 観念化（アイディエーション）：解決策に繋がりそうなアイディアを生み、発展させ、検証する。
(3) 実現化（インプリメンテーション）：市場投入までのプロセスを決定する。

各スペースから次のスペースへ進む間には、様々な調査・検証・整理などを行っていく。

着想から観念化へ進める間では、当該事業の問題点やチャンスとなる事項を整理し、人々の行動、思考様式、ニーズやウォンツを観察する。また、当該事業の制約（時間や資源の不足、貧弱な顧客基盤、市場の縮小など）を踏まえて、意見を交わし、ストーリーを練っていく。当該事業を取り巻く外部環境や内部環境を調査していく為、ITCのプロセスや手法では 5 フォースや 4P4C、SWOT 等の分析手法が適用できる。特にこの間での活動が大切であり、繰り返し行ったり来たりしながら情報を整理しブラッシュアップしていく。

観念化から実現化への間では、ストーリーのシナリオ固めをしていく。まず、スケッチから様々なシナリオを書き出し、アイディアを具現化する。さらに顧客を中心に置いて、これまでどのような経験をしたのか、どういうふうになりたいのか等について具体的に説明できるようにしていく。プロトタイピングと検証を繰り返していき、さらにストーリーを磨く。内部でコミュニケーションを図りつつ、さらに多くのプロトタイプを作成して、ユーザーテスト、内部テストを実施する。

実現化から着想の間では、市場での評判を広め、ビジョンの実現後においての検証を行い、次のビジネスへの糧とする。

2　創造性を取り戻すには

2.1　創造性を阻害する 4 つの恐れ

ほとんどの人間は生まれながらにして創造的であり、子供の頃は突拍子もない質問をしたり、空想の世界を語ったりして周囲の大人たちを驚かせる。しかし、学校や社会の中で育っていくうちに教育を受け社会性を身につけていくうちに、抑圧され用心深く慎重になる。そうすると、「創造的な人」と「創造的ではない人」に世の中が二分されているかのように感じられ、多くの人は自分が後者だと思いこんで諦めることが多い。

果たして、多くの大人たちは創造性を失ってしまったのだろうか。

創造性を発揮し、発送したものを発表する勇気は下記のような恐れに阻害される。

(1) 厄介な未知なるものへの恐れ
(2) 評価されることへの恐れ
(3) 第一歩を踏み出すことへの恐れ
(4) 制御できなくなることへの恐れ

2.2　小さな成功を積み重ねていく

4 つの恐れを克服し、新しい創造を行うにはどうすればよいだろうか。

課題を小さな段階に分け、ひとつずつ成し遂げることで自信を取り戻すアプローチが有効である。創造性を妨げる 4 つの恐れに対して、具体的にはそれぞれ下記のような小さな成功を積み重ねていく経験のプロセスを経ると良いだろう。

まず、厄介な未知なるものへの恐れに対しては、外の世界に出てニーズを見聞きすること、直にユーザーの声を聞くフィールドワークを行うことで漠然としていた「未知なるもの」が未知ではなくなっていくだろう。

評価されることへの恐れについては、自分で評価しないことの他、部下等にフィードバックを行う際には肯定的な言葉を投げかけるようにすること。そして、改善を促す助言をする場合にはあくまで個人的な提案であるという風に伝えると良い。

第一歩を踏み出すことへの恐れに対しては、いきなり大きな目標に向かっていくには大きな勇気が必要である為、大きな課題全体への照準をやめて、すぐに取り組める小さな断片から始めること、まずは一歩を踏み出してみることが肝要なので、計画や準備はそこそこに始めてみることなどが挙げられる。

制御できなくなることへの恐れに対しては、プロジェクトチームで協力し合って取り組んでいくことを念頭に、自分のアイディアに固執せず、他者の良いアイディアを受け入れる柔軟な姿勢を保つようにしていくと良いだろう。

このようにして恐れを克服し、イノベーションの道を進んでいこう。

3　創造的解決を導く方法

3.1　デザイン思考は社会技術

例えば TQM（総合的品質管理）活動では、かんばん方式や QC サークルなどの手法と「現場の作業員は、通常要求されているよりもはるかに高水準の仕事ができるだろう」という気づきが結びついて著しい改善を実現した。

このように、手法と考察が一体となって業務プロセスに反映された新しい仕事のやり方を、社会技術と見なすことができる。

デザイン思考は社会技術と言える。イノベーションの分野で、人材の創造性とコミットメントを十二分に引き出し、業務プロセスを劇的に改善する可能性を秘めている。

しかし、想像力の発揮を阻むものがある。それは、人間の先入観、現状への執着、特定の行動規範への思い入れ（しきたり）等である。デザイン思考はこれらを封じ込めることができる。

イノベーションの妨げになる様々な人間の傾向と、それに対抗するデザイン思考のツールと手順がどのように役立つかを見ていこう。

3.2　イノベーション成果とジレンマ

イノベーションが成功するには、「秀逸な解決策」「少ないリスクと変革コスト」「従業員の賛同」の 3 つのポイントが揃わなくてはならない。長年、産業界ではこれらの実現に奏功する戦術が考え出されてきたが、組織ではしばしば新たな障害や二律背反に遭うというジレンマを抱えている。

まず、秀逸な解決策の観点から見ていく。例えば課題を慣例的な方法で定義すると、もちろん従来型の自明な解決策しか出てこないことが多い。その一方で、趣向を変えた質問に対しては、その回答から独創的なアイディアの発見につながりやすくなる。た

だし、チームが際限なく課題を掘り下げる一方、先を急いでしまい、どの問いと向き合うべきかを十分に検討しなくなる恐れがある。

ユーザー主体の基準を取り入れることで大変優れた解決策が導出されることがよく起こる。それでヒアリングによる市場調査が行われるのだが、まだ存在しない製品やサービスについて「欲しいか」と聞かれても、ユーザーは答えづらい状況に陥ることがある。

多様な意見を取り入れることも、解決策の質的向上に効果がある。しかし、まったく異なる意見同士が出て険悪な状況となり、収拾つかなくなることがある。

少ないリスクと変革コストについて見てみると、イノベーションには不確実性が付き物であり、複数の選択肢を用意しがちである。しかし アイディアが多すぎて焦点やリソースが拡散する恐れがある。

最後に、従業員の賛同についてだが、もちろん支持を得るのは不可欠である。アイディア出しのプロセスに巻き込んでしまえば賛同を得やすいのだが、大勢、視点が異なる人々の意見を聞くとまとまらない。

3.3 デザイン思考のフェーズに沿った 7 つのステップ

デザイン思考は主に 3 つのフェーズと 7 つのステップを進んでいくフレームワークに当てはめることができる。四角四面な進め方ではイノベーションを起こせないように捉えられるかもしれないが、顧客への対面調査等を通して相手の考え方に深く寄り添いステークホルダーと共創していくチームマネージャにとって、慣れない経験に挑戦していく上で枠組みや一定の手順が安心感をもたらす。各段階で明確な成果が生まれ、次の段階ではそれが新たな成果につながる。この繰り返しを実践していくことで、実用的なイノベーションが実現するのである。

「顧客の発見」「アイディアの創造」「検証」の 3 つのフェーズと、7 つのステップを「イノベーターの問題点」「デザイン思考」「成果の向上」の視点でキーワードのみを図 3.1

にまとめた。

デザイン思考のフレームワークは、調査から実地展開への自然な流れを作るものである。顧客の立場から集めた情報を基準を設けて選考し、解決策へ向けて仮説を立ててプロトタイプを用いて検証することを繰り返し、イノベーションをさらに磨いてユーザーテスト等実環境での試行や実証実験を経て市場へと送り出す。そうしたプロセスを経ていくことによって、創造的解決策を導いていくのである。

	イノベーターの問題点	デザイン思考	成果の向上
顧客の発見			
	専門性・経験にとらわれる	顧客の立場に立つ	顧客理解の深化
	定性的データの量と煩雑さ	掘り下げる	新たなインサイトと可能性
	チームの足並み揃わない	調整する	利用者にとって本当に大切な事柄への注力
アイディアの創造			
	異質なアイディア氾濫	創発	少数多様な解決案
	有効/無効 従来の先入観	明確化	有意義な実験計画
検証			
	共通理解不足	先行体験	低コストで正確なフィードバック 解決策の真価の見極め
	不確実性からの恐れ	実地学習	新しい製品や戦略へ チーム一体で取り組む自信

図 3.1　デザイン思考のフェーズに沿った 7 つのステップ

参考文献

(1) 山田忠雄、柴田武、酒井憲二、倉持保男、山田明雄、上野善道、井島正博、笹原宏之 編、新明解国語辞典 第七版、三省堂、2011 年 12 月。
(2) ハーバード・ビジネス・レビュー編集部　編著、DIAMOND ハーバード・ビジネス・レビュー編集部　訳、ハーバード・ビジネス・レビュー　デザインシンキング論文ベスト 10　デザイン思考の教科書、ダイヤモンド社、2020 年 10 月。

第Ⅰ編　デザイン思考とは／創造的解決策を導くアプローチ

第Ⅱ編

イノベーションに関わるデザイン思考

新保康夫

第Ⅱ編　イノベーションに関わるデザイン思考

目次

1　リバース・イノベーション

1.1　リバース・イノベーションとは

リバース・イノベーション[1]とは、新興国で生まれた技術革新や経営革新などが、先進国に逆流して波及させることである。場合によっては大きな破壊力を持って先進国の市場を席巻する場合がある。

また、リバース・イノベーションは、国際経営論における技術革新等についての概念である。

1.2　5 つの落とし穴

リバース・イノベーションを語るとき、成功事例ではなく、失敗事例として日本の家電メーカーの白物家電について思い浮かべる人がいると思われる。

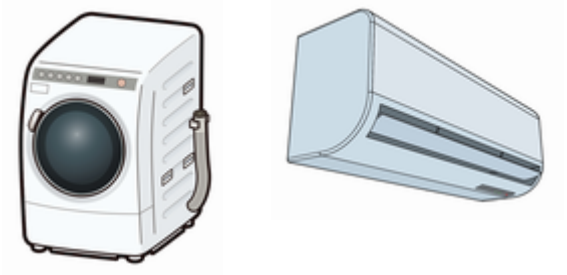


図 1.1　家電のイメージ

ここでは、失敗しやすい 5 つの落とし穴について、以下に挙げる。

(1) 市場セグメントを既存製品に合わせる
(2) 機能を省いて価格を下げる
(3) 新興国市場における技術要件すべてについて熟慮していない
(4) 利害関係者を顧みない
(5) 新興国市場向け製品がグローバルに訴求できると信じない

これら 5 つの落とし穴は、実際に落とし穴に落ちてしまいやすい項目である。

[1] Reverse innovation

1.2.1　市場セグメントを既存製品に合わせる

良い製品や商品だから、新興国市場でもそのまま通用するだろうと想定し、市場セグメントを製品に合わせてしまうことがある。残念ながら、新興国市場では、そのような市場セグメントが無い、もしくは、小さい規模であることが多い。結果、製品は売れない。最悪な場合は、経営者は現地での営業が悪いと評価してしまうことである。

1.2.2　機能を省いて価格を下げる

高機能や多機能の製品や商品が自国で売れるからといって、必ずしも新興国市場で売れるとは限らない。新興国市場にあった価格が必要である。ここまでの考えは正しいが、だからと言って、闇雲に、機能を省いて新興国市場にあった価格にしても売れるはずがない。新興国市場のニーズにあつた機能を省けば、魅力の無い製品となってしまう。

1.2.3　新興国市場における技術要件すべてについて熟慮していない

新興国市場へ参入する場合は、新興国の社会環境を含めた技術要件について十分な調査や分析を行わずに参入すると失敗する場合がある。例えば、電力状態について十分な調査を行わないと停電が当たり前という場合がある。このようなところに、電気が必要な製品は持ち込めない。

1.2.4　利害関係者を顧みない

直接的な利害関係者については考慮するが、間接的な利害関係者や製品を販売や保守する利害関係者まで考慮しない場合がある。製品を消費者に購入してもらうためには、現地の販売員や保守する人が必要である。故障しても修理できない製品や交換してもらえない商品では、新興国市場でも購入してくれないであろう。

1.2.5　新興国市場向け製品がグローバルに訴求できると信じない

新興国市場向け製品は自国や他の国では売れないであろうと考えている場合がある。確かに。そのままであれば無理ではあるが、自国や他の国の仕様に合っていれば売れないであろうか。せっかくの大きなビジネスチャンスを逃している。

1.3　5 つの落とし穴を回避する設計の原則

前節の 5 つの落とし穴に対して、デザイン思考によるどのような設計の原則によって回避するのかを見てみよう。

それぞれの落とし穴に対応した 5 つの設計の原則がある。

(1) 解決策から離れて、問題を定義する
(2) 新興国市場だから得られる自由度の高い設計を使って、骨抜きの解決策でなく、最適な解決策を見つけ出す
(3) 消費者の問題の背後にある技術的状況を分析する
(4) なるべく多くの利害関係者を交えて製品をテストする
(5) 新興国市場の制約条件を活かしてグローバルで勝てるものを創り出す

1.3.1　解決策から離れて、問題を定義する

既存製品を販売するという目的の解決策から離れて、新興国市場セグメントに存在する課題や問題から定義するということが大切である。

例えば、農業用水としての灌漑(かんがい)用ポンプを販売することを考えて見よう。新興国では、電力事情が良くないので、電線による電力供給のポンプを販売するのは難しい。

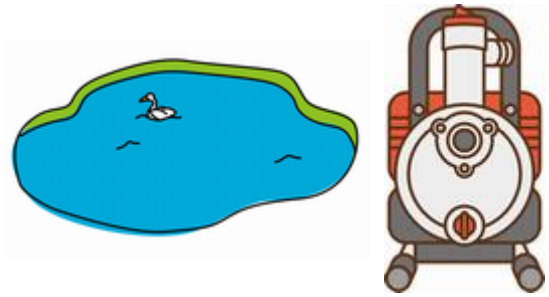


図 1.2　灌漑用ポンプのイメージ

電力で動く方が安定していて、燃料などを補給しなくても良い利点がある。設置する場所では、晴天の日が多く、日照時間が長い場合、太陽光発電と蓄電池による電力供給を行えば、比較的安定した電力を供給でき、現地でのポンプの通常利用が問題がなければ、そのような構成でポンプを販売すれば売れる可能性があるということになる。

このように、既存の製品ポートフォリオの外側にあるチャンスを見極めることにより、課題が解決することがある。

図 1.3　太陽光発電のイメージ

1.3.2　新興国市場だから得られる自由度の高い設計を使って、最適な解決策を見つけ出す

新興国市場というだけで、自国で販売している製品の機能を省いて安価な製品を提供するよりも、自由な発想で新興国市場だから得られるメリットを活かしたデザインやプランを用いて、新興国にフィットした製品を提供することが必要である。

例えば、新興国では労働単価が安いため、人手による製品生産の方が機器を導入した生産よりコストが低く抑えられる。このため、機能をそれほど省かない製品が提供できるため、市場でも十分な競争力を持った製品となる。さらに、人による生産のため、保守も比較的容易に対応できる。

図 1.4　コストの比較

このように、新興国市場にはいろいろな制約条件が多いが、それらを考慮した新興国ならではの自由度の高い設計が可能である。また、新興国では自由度もさまざまな形を取る。

1.3.3　消費者の問題の背後にある技術的状況を分析する

根底にある技術的状況は、新興国では著しく異なって見える。都市部と農村部、メイン通りと裏通り、中心部と周辺部など新興国の社会状況や環境を考慮して、消費者だけを見つめるのではなく、消費者の問題の背後にある技術的状況を分析することが大切である。例えば、トラクターを考えて見よう。トラクターは、本来、農地を耕すものである。しかし、交通機関が発展していない新興国では、トラクターは人を運ぶものかもしれない。人を乗せる台車を繋げる仕掛けを付けておいた方が製品としては売れるであろう。

図 1.5　トラクターのイメージ

1.3.4　なるべく多くの利害関係者を交えて製品をテストする

提供する製品や商品に目を向け、最終消費者だけを考えてしまいがちだ。製品へのアプローチも最終消費者だけの意見を反映させてしまいやすい。新興国市場では、なるべく多くの利害関係者を交えて製品をテストすることが重要である。そのためには、設計プロセスの最初に、製品の成功を左右する一連のすべての利害関係者を洗い出すことである。まずは、誰がエンドユーザーか、どのようなニーズか把握し、次に、誰がその製品の生産、流通、販売、支払い、修理、廃棄するか考慮すべきである。現地でのそれぞれの意見を把握し、シミュレーションを行い、評価することである。

1.3.5　新興国市場の制約条件を活かしてグローバルで勝てるものを創り出す

新興国市場の製品は、自国や先進国には利用出来ないと、初めから決めつけない方がよい。実は、グローバルに売れる要素が隠されている場合が多い。そのためには、新興国市場の制約条件を活かしてグローバルで勝てるものを創り出すことである。まずは、解決策を計画する前に、新製品やサービスに影響を及ぼす固有の制約条件を確認する。次に、価格、耐久性、素材など新しい設計で満たすべき要件が確定することである。例えば、洗濯機で考えよう。環境の良くない新興国では、洗濯機の設置場所や水質が非常に悪い。天候は雨期や乾期があり、購入者層の生活水準は先進国より低く家族構成が多い場合、一槽式の全自動洗濯機より二槽式の洗濯機の方が良い。

かつ、劣悪環境に壊れにくいものが要求される。新興国市場で受け入れられる製品を提供し、先進国に求められるデザインや安全性を考慮した洗濯機を製造すれば、先進国での土木・建築現場での洗濯機として利用ができる。また、高齢者には多機能な洗濯機より簡単な洗濯機の方が使い易いと好まれる場合がある。昨今のシンプル思考の観点から二槽式洗濯機が受け入れられる可能性がある。

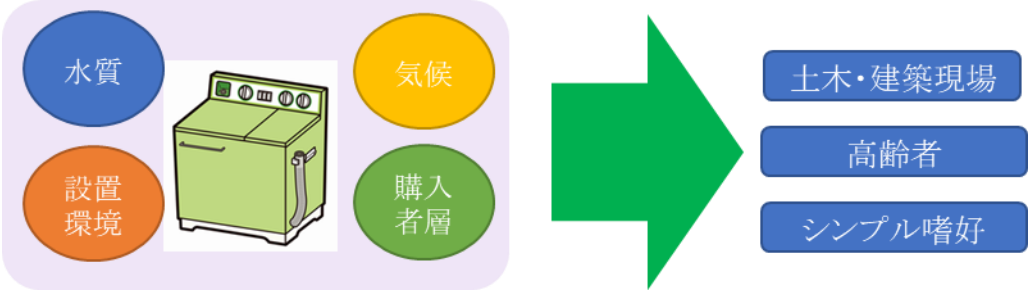


図 1.6　洗濯機の例

2　イノベーション・カタリスト

2.1　イノベーション・カタリストとは

2.1.1　カタリスト

カタリスト[2]とは、化学反応を促す「触媒」を表す英語であり、金融市場では相場を動かす契機となる材料とされ、システム開発においては、ウェブアプリケーションフレームワークのひとつとされている。

2.1.2　イノベーション・カタリストの定義

イノベーション・カタリストを明確に定義として示されているのは少ない。ここでは、イノベーション・カタリストとは、イノベーションを触発し、全社に改革を推進するマネジャーと定義する。イノベーション・カタリストは、横断する知見を持ち、ユーザー視点でコンセプトを生み出しながらビジネスを推進していくビジネスプロデューサーからリサーチャー、デザイナーである。

一般的には、1人ですべての役割をこなせるスーパーマンは難しいので、複数の人材でチームとして活動することになる。

2.2　イノベーションへのステップ

2.2.1　イノベーションを成功させるための 4 つのポイント

イノベーション・カタリストがいたからと言って、イノベーションが行える訳ではない。経営者の意気込み、お金の投入が必要となる。また、従業員のモチベーションも当然必要だ。中小企業においては、イノベーション・カタリストとなる人材が見受けられない場合もあるので、その場合は、外部から人材を投入し、イノベーション・カタリストの育成とイノベーションへのサポートが必要となってくる。

ここまでが、イノベーションのスタートラインであり、成功するためには以下の 4 つのポイント(もしくは、ステップ)が重要である。

(1)　草の根から改革を始める

---

[2] Catalyst

(2) アイデアを実践に移す
(3) プレゼンテーションより実践を重視
(4) イノベーションを起こし企業文化を改革する

各項目の説明については、次に説明する。

2.2.2 草の根から改革を始める

中小企業の経営者がスティーブ・ジョブズであれば、経営者自らイノベーション・カタリストとして経営トップからイノベーションを起こすことができる。しかしながら、すべての人が誰でもスティーブ・ジョブズにはなることは難しい。多くの場合は無理である。

そのためには、いくつかのツール、コーチング、研修により、イノベーション・カタリストを育成するとともに、現場の力でイノベーションを生み出し、顧客を感動させるように現場の１人１人から意識変革を草の根から始めることが大切である。当然ながら、経営層も自らイノベーションの海に船出する自覚と行動が求められる。

図 2.1　現場から草の根改革のイメージ

2.2.3 アイデアを実践に移す

アイデアだけでは、イノベーションにはならない。実践に移すことでアイデアがイノベーションのトリガーとなる。実践といっても、直接、現業を変革することではない。例えば、ワーキンググループの中で、アイデアを現業へ変革をシミュレーションすることにより、課題や制約条件などが洗い出される。さらに、プロトタイプや実験により、より実務に近い形で変革を試すことができる。このとき、顧客の一部を参加してもらうことにより、顧客の反応や意見から学習し、それらを反映することで、より確実なイノベーションとなる。

この時、イノベーション・カタリストは、デザイン思考のコーチ役として、現場メンバやプロジェクトメンバーをサポートしていくことが大切である。

図 2.2　アイデアを実践に移す

2.2.4　プレゼンテーションより実践を重視

イノベーションを行うと、良いと思ってしまうアイデアや分析をしてしまうとプレゼンテーションで満足してしまい、実践が先細り、「こんなはずじゃなかった」という結果になってしまう場合がある。プレゼンテーションも必要ではあるがチームや関係者との意識や段取りの整合性という点にとどめ、イノベーションは成功するための実践を重視する方が大切である。そのためには、実験を通じてみずから顧客に学ぶことが重要である。さらに、そこから問題の発見（ペイン・ストーム）プロセスを通して、顧客の抱える悩みの種（ペイン・ポイント）を見出し、解決策を提供することである。

図 2.3　実践重視のイメージ

2.2.5　イノベーションを起こし企業文化を改革する

最初のイノベーションは、選抜されたメンバからかもしれないが、このイノベーションを起こして企業文化を変革し、社内全体からイノベーションが生まれる企業になることが大切である。この時には、イノベーション・カタリストの人を拡充し、社内の対象となる業務や部署の範囲の拡大し、企業全体、もしくは、グループ企業全体に範囲を広げることが出来れば良いであろう。そのためには、イノベーション・カタリストがさらに重要となるため、彼らの価値を高める必要がある。

3　振り返り

3.1　リバース・イノベーション

リバース・イノベーションのデザイン思考の原則は、通常ビジネスでも大切である。ターゲット対象の所得、業種などが異なれば、同様なことがいえるかもしれない。

このため、ビジネスの場（プレイス）での事業ドメイン分析や顧客分析が重要と考える。

リバース・イノベーションには、2 つのデザイン思考が必要かもしれない。一つは、設計を行うと言う意味でのビジネスを考えるデザイン思考であり、PDCAサイクルも必要と言える。もうひとつは、分析ツールなどのデザインを利用するということである。俗に言う「車輪の発明」にならないようにすることも大切である。

3.2　イノベーション・カタリスト

イノベーション・カタリストの導入による変革は、「リーン・スタートアップ」との共通点が多いかもしれない。その理由として、以下の 3 点が挙げられる。

(1) 長大な計画策定よりもアイデアの実践重視
(2) 小さく始めて、大きくしていく
(3) PDCAサイクルは大切

さらに、デザイン思考のツールとしては、「ビジネスモデル・ジェネレーション」の導入事例で学ぶビジネスモデルのつくり方やキャンバスの導入・活用のための手順やツールは、実践でのツールとして参考になるかもしれない。

3.3　デザイン思考とイノベーション

今回、イノベーションという観点からデザイン思考を見ているが、デザイン思考の意味であるデザイナーが思考するプロセスを用いて行い、「設計する」ということの重要性が経営や事業に変革をもたらすときには大切であると考える。

また、IT におけるシステム開発やプログラム開発の思想や手法に似ていると思われるところがある。

(1) アジャイルと何か共通点がありそう
　リーン・スタートアップとアジャイル開発とでは共通する部分が多い。
　このため、イノベーション・カタリストでリーン・スタートアップの共通点で挙げた 3 つの理由がアジャイルにもあてはまると考える。

(2) ツールの活用
イノベーションにおいては、手法やツールを上手く使うことが求められる。システム開発においても、手法やツールを上手く使うことにより、品質と時間を改善できる。また、ビジネスモデル・ジェネレーション ビジネスモデル設計書に掲載されている手法やツールが利用できるのではないかと考える。

(3) デザインパターン
デザイン思考ということから、デザインパターンというものがあるのではないかと漠然とだが想定する。例えば、システム開発やプログラム開発でのオブジェクト指向で常識である「GoFのデザインパターン」ではないけれど、手法やツールの組み合わせ方があるのかもしれない。

(4) コンポーネントカタリシス
システム開発やプログラム開発でのオブジェクト指向でのコンポーネントベース開発ではないが、コンポーネントのように思考の組み合わせがあるのかもしれない。例えば、ITCのIT経営推進プロセスガイドラインのようなものかもしれない。

(5) 車輪の開発にならないように
イノベーションという言葉から、何か新しい考えや新しい手法というものが必要と錯覚してしまうと、既に、先人達が考えた考え方や手法を無駄に考えてしまうことは、イノベーションにおける時間の無駄となり、ビジネス機会を失うことにもなりかねない。

最後に、リバース・イノベーションの 5 つの落とし穴への設計の原則は、実際上の経営課題の解決にも言えるのではないだろうか。さらに、イノベーション・カタリストは、変化の時代には必要かもしれない。そして、プロジェクトとしてのひとつの進め方にもなる。これらを通じて、デザイン思考はナレッジの基本かもしれない。

参考

リーン・スタートアップ

コストをそれほどかけずに基本的な製品や、基本的なサービス、基本的な機能を持った試作品を短期間で作り、顧客に提供することで顧客の反応を観察する。

その観察結果を分析し、製品、サービスが市場に受け入れられるか否か判断し（市場価値が無ければ撤退も考慮）、試作品やサービスに改善を施し、機能などを追加して再び顧客に提供する。

ビジネスモデル・ジェネレーション

コアとなるツール「ビジネスモデル・キャンバス」を使って、新しいサービスやソリューションをビジネス化していくための考え方や、既存のビジネスモデルにイノベーションを起こすためのプロセスを身につけるメソッドである。

図 3.1　ビジネスモデル・キャンバス

参考文献

(1) ハーバード・ビジネス・レビュー編集部　編著、DIAMOND ハーバード・ビジネス・レビュー編集部　訳、ハーバード・ビジネス・レビュー　デザインシンキング論文ベスト 10　デザイン思考の教科書、ダイヤモンド社、2020 年 10 月。
(2) エリック・リース　著、井口　耕二　訳、伊藤　穣一(MIT メディアラボ所長) 解説、リーン・スタートアップ　ムダのない起業プロセスでイノベーションを生みだす、日経 BP 社、2012 年 4 月。
(3) 2012 年度ITC大阪城　テーマ研究調査活動 「リーンスタートアップによる ITC ビジネスへの考察」、https://www.itc-osakajo.jp/?page_id=18。
(4) アレックス・オスターワルダー　著、イヴ・ピニュール　著、小山　龍介　翻訳、ビジネスモデル・ジェネレーション ビジネスモデル設計書、翔泳社、2012 年 2 月。
(5) 2013 年度ITC大阪城テーマ研究調査活動 「ビジネスモデル・ジェネレーション研究」、https://www.itc-osakajo.jp/?page_id=18。
(6) 特定非営利活動法人 IT コーディネータ協会、IT経営推進プロセスガイドライン Ver.3.1、2018 年 5 月、https://www.itc.or.jp/about/guideline/#pgl3。
(7) Alan Cameron D'Souza, Desmond Francis Wills 著、Objects, Components, and Frameworks with UML: The Catalysis Approach 、Addison-Wesley Professional、1998 年 10 月。
(8) エリック ガンマ,ラルフ ジョンソン,リチャード ヘルム,ジョン ブリシディース 著、オブジェクト指向における再利用のためのデザインパターン、翔泳社、1999 年 10 月。

第Ⅱ編　イノベーションに関わるデザイン思考

第Ⅲ編

情報にダマされるしくみ（情報を正しく認識するために）

脇阪公昭

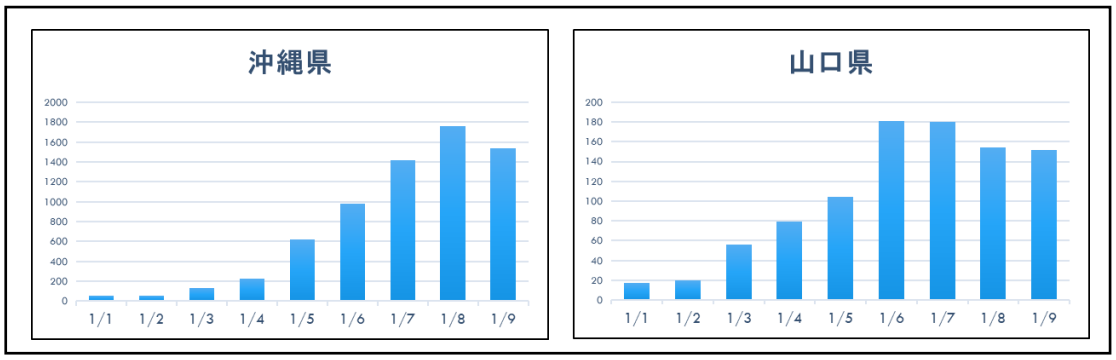
第Ⅲ編　情報にダマされるしくみ（情報を正しく認識するために）

目次

1　はじめに

まず、このグラフを見ていただきたい。これは NHK の特設サイト 新型コロナウィルスから、2022 年 1 月 10 日時点においてまん延防止重点措置が発令された沖縄県と山口県の日別感染者数のデータをグラフ化したものである。

[1]図 1.1　日別新型コロナ感染者数（2022 年 1 月 10 日）



このグラフを見て、どのような印象を受けただろうか。

このグラフについては後ほど解説するが、人は過去の経験や置かれている立場、先入観などによって、情報を誤って認識してしまうことがある。更に情報の表現によっては、その傾向が強くなる。

本報告書では、情報を正しく理解するために、そのことを阻害する「認知バイアス」と、数値やグラフなどの表現について記す。

[1] （参考）NHK　特設サイト新型コロナウィルス

2　正しい理解を阻害する認知バイアス

人は、物事を判断する際に、今までの経験や、思い込み、先入観、固定概念などに影響を受け、合理的ではない意思決定や行動をとってしまうことがある。このような心のゆがみが生じてしまう現象のことを、認知バイアス（cognitive bias）と呼ぶ。認知バイアスは心理学用語のひとつであり、数多く存在する。

人がもつ認知バイアスによる心のゆがみを認識していないと、情報を正しく理解し把握できないことがある。本報告書では、数多い中から比較的良く聞く5つの認知バイアスを紹介する。

2.1　確証バイアス

人は自分の考えや仮説が正しいことを確認するために、都合の良い情報ばかり集め、そうでない情報は無視、又は集めようとしない傾向がある。このような傾向を確証バイアスという。

例えば、あなたがある商品の購入を検討していて2つの商品に絞り込んだ結果、A 商品の方が B 商品より優れていると考えたとする。そして店舗に行って両商品の説明を聞いた時に、A 商品については良いところばかりを聞き、B 商品については劣っているところのみに耳を傾けられた経験はないだろうか。もしそうであるならば、確証バイアスがかかっていた可能性がある。血液型が A 型の人はきっちりしていると思い込み、几帳面な部分ばかりを探してしまうのも、確証バイアスの例としてよく挙げられている。

2.2　ハロー効果

ある特定のもののどこか優れている点を見つけると、関係のないことにも優れていると考えてしまう傾向がある。逆に、劣っている点を見つけると、他のものまで劣っていると考えてしまうこともある。これを、ハロー効果（逆ハロー効果）という。

私たちは、すべてのことを把握していないと、わからない部分も同じように評価してしまう傾向がある。例えばあなたのオフィスに、さわやかなで穏やかそうな女性がいた場合、話をしたことがなくても、きっと仕事ができて優しそうな女性と思ったことはなかっただろうか。

2.3　バンドワゴン効果

複数の選択肢がある場合に、たくさんの人に支持されていると、自分の判断よりも正確であると思い込んでしまう傾向がある。その結果、その選択をとろうとする人が増え、更に強くなるという効果がある。これを、バンドワゴン効果という。

バンドワゴンとは、パレードなどで行列の先頭に位置する楽隊車のことであり、「バンドワゴンに乗る」という表現は、時流に乗る、勝ち馬に乗るという意味で使われる。

例えば、多くの人が並んでいる飲食店があった場合、この店はきっとおいしいだろうと思って行列に並ぶのも、マーケッティングの現場で見られるバンドワゴン効果のひとつである。

2.4　現状維持バイアス

何か変化させることによって、より良くなる可能性があるのに、変化によって生じる損失を恐れて、現状維持を望む傾向がある。これを、現状維持バイアスという。

失敗しない傾向を好む特性のもつ日本人は、この傾向が強いといわれている。自分から初期設定しなければ自動的に選択・設定されるデフォルトの効果は、現状維持バイアスに関連したテクニックのひとつである。

2.5　正常性バイアス

人は、何か危険や異常な事態に直面した際に、「自分だけは大丈夫だ」と考えたり、正常な範囲だと考えたりする傾向がある。これを、正常性バイアスという。

予期しない事態が自分にふりかかったとしても、自分は大丈夫と考えることで、ストレスから心身を守る防御作用が関係している。

あなたが朝起きた時、体がだるくて熱っぽいように感じても、これは風邪であってコロナではない（自分は大丈夫）と考えたりしないだろうか。台風が来ているにも関わらず、自分は大丈夫だと考え、海に船を見に行ったり、畑の作物を見に行ったりして災害が発生するのも、この正常性バイオスが一因となっていると思われる。

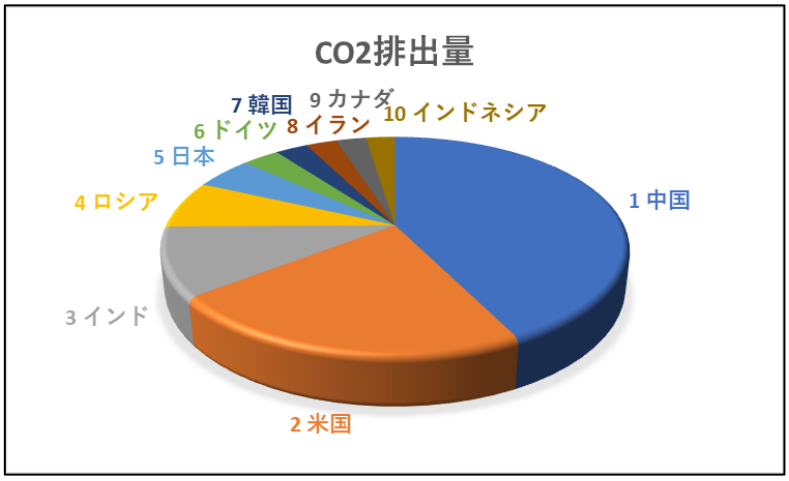
3　正しい理解を阻害する表現

正しい数値を基に作られたグラフ、図などであっても、表現のしかたによって誤った理解をしてしまうことがある。特に情報発信者が、前述の認知バイアスを故意に利用すると、正しく理解することがより難しくなる。

以下にいくつかの例を紹介する。

3.1　（例 1）3-D グラフ

下記グラフは、外務省から公開された 2018 年度における各種エネルギーの利用時に発生した CO2 の排出量の TOP10 である。このグラフを見ると、1 位の中国と 2 位の米国の CO2 排出量はさほど違わないように見えないだろうか。



[2]図 3.1 ［3-D］ 二酸化炭素（CO2）排出量の多い国（2018 年）

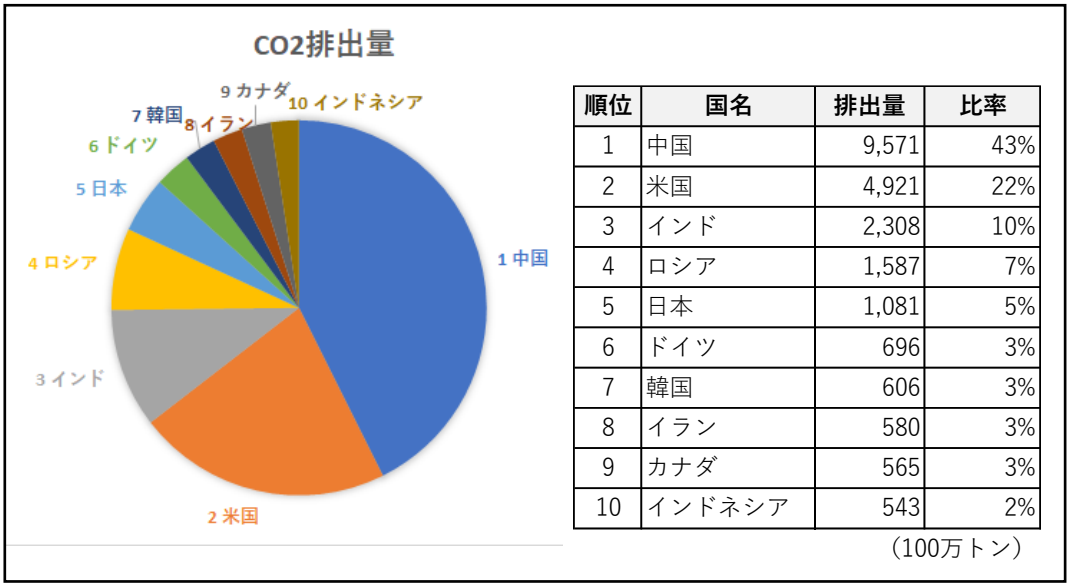
このグラフの基となった数値を見ると、中国の CO2 排出量は約 95 億トン（全体比：43%）、米国は約 49 億トン（全体比：22%）となっている。すなわち、米国は中国の約 1/2 の排出量であるが、このグラフからそのように伝わっただろうか。

このグラフは、何らウソはついていない。しかし情報を見る人の立場によっては、自分に都合のいい情報ばかり集める「確証バイアス」により、「中国と米国の CO2 排出量は同程度である」と判断してしまうかもしれない。

このグラフを 2-D グラフに変えると、見え方も大きく変わってくる。

---

[2]　（参考）外務省　二酸化炭素（CO2）排出量の多い国（2018 年）



順位	国名	排出量	比率
1	中国	9,571	43%
2	米国	4,921	22%
3	インド	2,308	10%
4	ロシア	1,587	7%
5	日本	1,081	5%
6	ドイツ	696	3%
7	韓国	606	3%
8	イラン	580	3%
9	カナダ	565	3%
10	インドネシア	543	2%

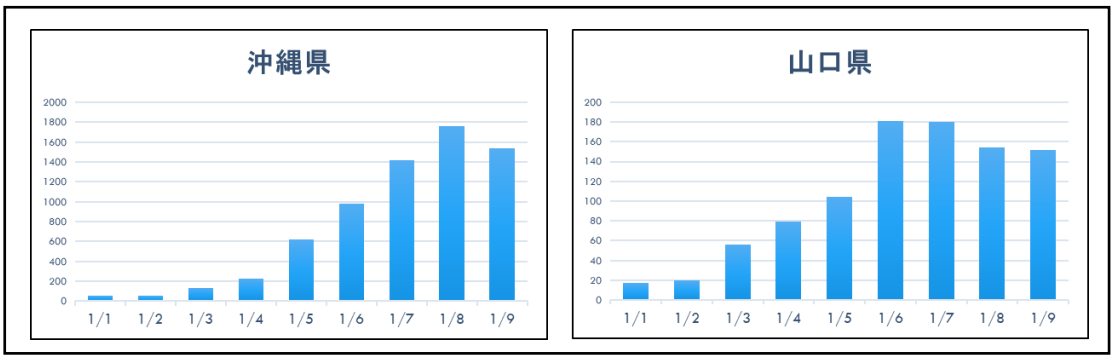
（100万トン）

[2]図 3.2 ［2-D］ 二酸化炭素（CO2）排出量の多い国（2018 年）

3-D グラフは、見た目は良いが、情報を誤って伝える可能性があるため、発する側も受ける側も十分に注意する必要がある。

3.2 （例 2）目盛りの異なるグラフ

次は、本報告書の最初に記載したグラフを再度見て頂きたい。



[3]図 3.3　日別新型コロナ感染者数-1（2022 年 1 月 10 日）

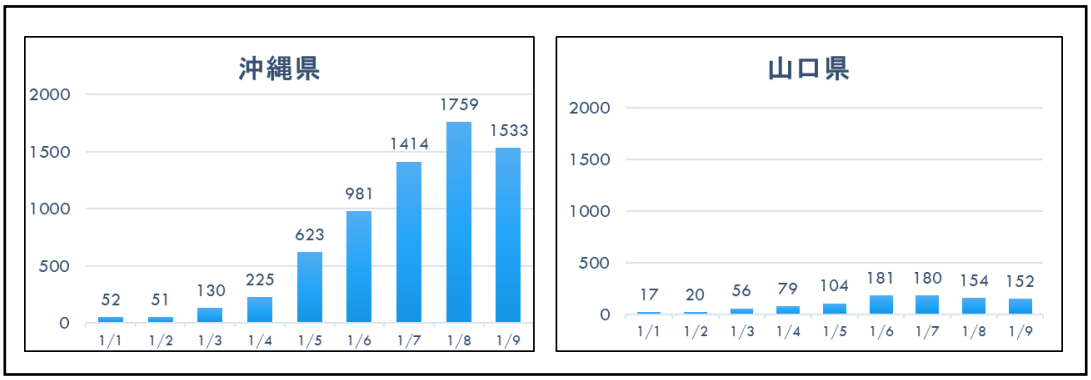
このグラフを見ると、沖縄県と山口県は同じような感染者数となっているように見えないだろうか。

縦軸の目盛りに注目していただきたい。注意深い方はお気づきかと思うが、2つのグ

[3]（参考）NHK　特設サイト新型コロナウィルス

ラフの目盛りは全く異なっている。このグラフは、故意に目盛りを小さく表示しているが、見る人に何らかのバイアスがかかっていた場合、誤った認識をしてしまう恐れがある。

この2つグラフの目盛りを合わせると下記のようになり、感染者数は全く異なっていることが明確になる。



[3]図 3.4　日別新型コロナ感染者数-2（2022 年 1 月 10 日）

3.3 （例 3）キャッチコピー

あなたがある保険に加入していて、契約期間が切れる頃に下記のような広告が送られてくるとどのように感じるだろうか。

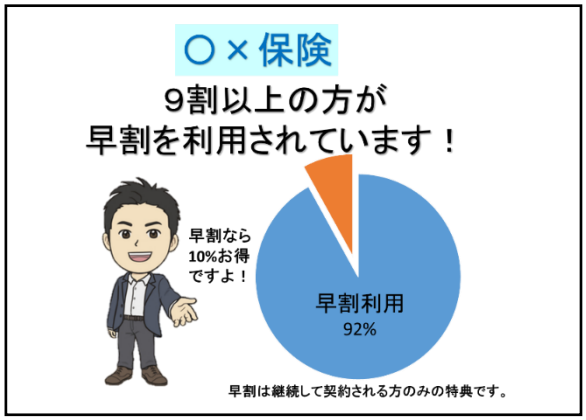


図 3.5　保険の広告例

「早割にすると、10%も安く契約できるのか」「ほとんどの方が早割で契約しているのか」「この得点は継続契約者のみの特典なんだな」「早割はいつまでかな」と興味をもち、広告に出ている人物が誠実な人柄の有名人であれば、更に前向きに検討するのではないだろうか。

この広告は、多数の人が選択していることを演出した「バンドワゴン効果」や、有名人

のイメージを利用した「ハロー効果」、変化によって生じる損失を恐れる「現状維持バイアス」などを利用している。

この広告には、何らウソはない。発信者が効果的に認知バイアスを利用した例である。しかし、広告を見た人はそのことを認識していたであろうか。

4 まとめ

プレゼンテーションや広告などには発信者の目的がある。その目的を達成させるために、認知バイアスを利用したものも少なくない。人は認知バイアスという心のゆがみがあるということを知った上で、情報を読み取ることが大切である。

5 考察

今回、「ダマすプレゼンのしくみ」（佐々木幹夫著）などを読んで、「認知バイアス」を知り、「認知バイアス」をもつ人への関心を深めることができた。

人にはそれぞれ取り巻く環境や過去の経験、思い込みなどをもち、同じものを見ても受け止め方や解釈が異なり、結果として違った判断やアクションをとってしまうことがある。だからと言って「認知バイアス」が「悪」ということではなく、人は「認知バイアス」という心のゆがみをもっているのだということを知った上で、正しく情報を理解することが大切であろう。

このことを知った時、私はある書籍を思い出した。 その書籍は「反応しない練習」（草薙龍瞬著）であり、感情でムダに反応することによって生じる悩みについて書かれた本である。この書籍を思い出したのは、感情で反応することによって生じる悩みと、認知バイアスによって反応して生じる誤った理解に、共通的なイメージをもったためである。この書籍には、このように書かれている。

「反応せずに、まず理解せよ」
「心の半分を前に、もう半分を後ろにつかえ」

これは、前を見る心と、心の内側を見る心をもつことによって、ムダな反応をしないようにすることとして述べられているが、認知バイアスによってゆがんだ心の反応を、後ろの心で客観的、論理的に、かつ冷静に見る心を持てば、誤った情報の認識を少しでも防ぐことが出来るのではと考えている。

参考文献

(1) 佐々木幹夫　著、ダマすプレゼンのしくみ　数値・グラフ・話術・構成に隠された欺く手法とその見破り方、翔泳社、2021 年 6 月。
(2) 情報文化研究所（山﨑紗紀子／宮代こずゑ／菊池由希子）　著、高橋昌一郎監修、情報を正しく選択するための認知バイアス事典、フォレスト出版、2021 年 4 月。
(3) 草薙龍瞬　著、反応しない練習　あらゆる悩みが消えていくブッダの超・合理的な「考え方」、KADOKAWA、2015 年 7 月。

第Ⅳ編

モブプログラミング・ベストプラクティス

岡田誠司

第Ⅳ編　モブプログラミング・ベストプラクティス

目次

1 モブプログラミングについて

1. 1　モブプログラミングの定義

モブプログラミングとは、3 人以上の人々が 1 台のコンピューターの前に座って協力しながら問題を解決していくことで、分散知識の考え方をプログラミングに活かしたものである。

分散知識とは、人間の集団が持っている問題解決に役立てられる知識の総体のことだ。モブプログラミングはそれらの人々を集め、1 台のコンピューターの前に座らせるのである。

モブプログラミングの名称については、Woody Zuill がユーザーグループやカンファレンスでコーディング道場といったグループプログラミング活動をするときに、ちょっとユーモアを込めて「モブプログラミング」という言葉を使うようになった。Woody は、それらのイベントに参加した人々に対し、自分たちが使っている方法はペアプログラミングに似ているけれども、もっと多くの人々が群れ（モブ）をなして取り組むものだと説明した。その後、彼は基本的なルールを守りさえすれば誰もが貢献できるのがモブプログラミングであり、統制の取れない群衆（モブ）というわけではないと説明している。

1.2 モビングとペアリング

モブプログラミングはペアプログラミングから発展したものである。
ペアプログラミングとモブプログラミングの違いをケース別に3つあげる。

(1) まずペアプログラミングを試した方がよいのか？
(2) ペアプログラミングを導入済みだが、モビングに切り替えるべきなのか？
(3) ペアリングは好きではないが、モビングも同じじゃないの？

1.2.1 まずペアプログラミングを試した方がよいのか？

これは決してマストではない。多くのチームがペアプログラミングを省略し、まっすぐモブプログラミングに進んで成功している。両方の方法に習熟する事には確かにメリットがあるが、まずペアプログラミングをマスターしなければならないということはないし、ペアプログラミングに参加してみなければならないということさえない。それどころか、いきなりモビングを始めるメリットというものさえある。

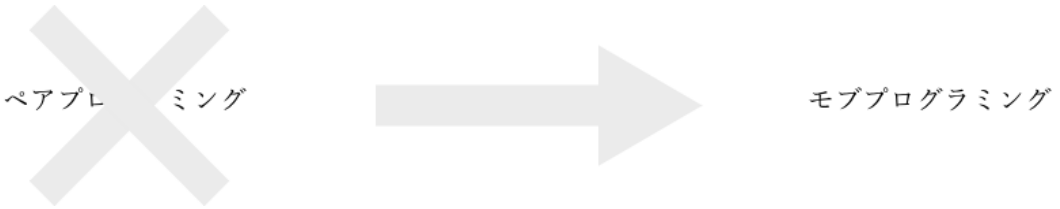


図 1.1　ペアプログラミングとモブプログラミング

1.2.2 ペアプログラミングを導入済みだが、モビングに切り替えるべきなのか？

まず同じ仕事でもひとりよりペアの方が早く問題を解決できることを示す学術研究が複数出ている。

それに対し、モブプログラミングとペアプログラミングを直接比較する研究成果はまだでていないが、アメリカ心理学会の 2006 年の研究によれば、複雑な問題の解決では、3 人、4 人、あるいは 5 人いた方がふたり（そしてひとり）よりもよいとされている。複雑な問題に挑むときには、多くの人々が共通の目標に向かってアイデアを出し合うモブの方がペアよりも優れている。

また、一般にモブはペアよりも中断しにくい。ペアプログラミングの場合、相方に電話がかかったり、会議の時間がやってきたり、その他どんな理由でも相方が中座しなければならなくなると、作業はストップしてしまう。私はずっとこういう中断に悩まされ、中断を減らすさまざまな方法を試してきた。そのなかでもっとも成功したのは、後述のポモドーロテクニックである。

ポモドーロテクニック

ポモドーロテクニックは、ひとりでもふたりでも使える生産性向上のためのテクニックである。1 日はポモドーロに分割される。1 ポモドーロは 25 分で、その間はひとつのことに集中する。

ペアプログラミングでポモドーロテクニックを使う場合には、電話、メール、臨時会議といった中断要因は、25 分間の個々のポモドーロからすべてシャットアウトされる。25 分が経過すると、5 分の休憩に入る。そのときには、膝を伸ばしたり、メールを見たり、トイレに行ったりしていい。5 分が過ぎると、タイマーを 25 分にセットし直し、新たなポモドーロに入る。このようにして同じサイクルが繰り返される。

ペアプログラミングでも、ポモドーロテクニックを使えば、細かい中断の多くは取り除ける。しかし、会議、病気、その他の理由で相方がいないときにどうすればよいかという大きな問題には対処できない。

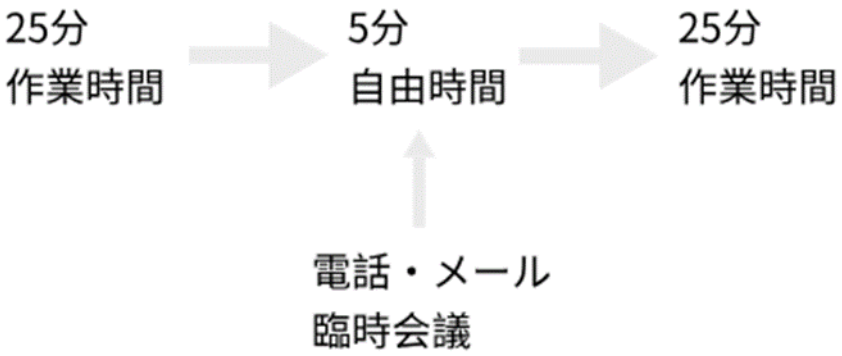


図 1.2 ポモドーロテクニック

1.2.3 ペアリングは好きではないが、モビングも同じじゃないの？

ペアプログラミングは、誰もが気に入る方法ではない。残念ながらペアリングとモビングには共通点が多いためペアリングが嫌いな人々はモブプログラミングにも同じ感想を持つ可能性があり、彼らは試してみる価値があるかどうかさえ疑うのである。

一旦モビングはペアリングとは異なる。それでもうやっぱりモビングは好きになれないと思うかもしれないが、ペアリングはともかくモビングは好きだと思う可能性も十分にある。

それはなぜかというとモビングの場合、共同作業者との関わり方が異なる。そのため関わりの強度が異なり、準備が異なる。これらは両者の違いのほんの一部に過ぎない。だから、ペアリングは大嫌いでもモビングは気に入るということは大いにあり得る。

図 1.3　モブプログラミングのイメージ

1.3 モビングの他人へのメリットの説明方法

1.3.1 リソース効率とフロー効率

モブプログラミングの考え方は、本質的にフロー重視であり、機能を安く作るのではなく、早く完成させることを目標としている。言い換えれば、リソース効率ではなくフロー効率を上げようという考え方だ。

リソース効率を重視すると、特定のタスクのスキルが高い人が生まれる。つまり、エキスパートが誕生する。

同じ報酬を与えているAとBのふたりの開発者がいたとする。Aはフロントエンドの仕事が得意で、Bの半分の時間で仕事をこなせる。このことを知っている管理職は、フロントエンドの仕事が必要になるたびに、それを早くこなせるAに仕事を振る。これはリソース効率を重視した仕事のしかたである。

この方法の問題点は、そのうちにAがフロントエンドのスペシャリストになり、Bはそうならないことである。こう言うと、別の問題があるようには思えないかもしれない。しかし、急にフロントエンドの仕事が増えてAだけではこなせなくなったときにはどうなるだろうか。フロントエンドの仕事のところで遅れが出て、全体のボトルネックになってしまう。

これと対照的なのが、フロー効率を上げる方法だ。フロー効率とは、機能全体を早く市場に送り出すことを重視する考え方である。フロー効率をあげようとするチームは、一丸となって新機能に取り組む。誰かが 1 日とか 1 週間（あるいは 2 週間でも）仕事から離れなければならなくなっても、チームは仕事を続けられる。ひとりいない分、確かにペースは少し落ちるかもしれないが、新機能のリリースに向けた作業は続く。

新機能の開発コストを下げるよりも、新機能を早く市場に送り出した方が大きな利益が得られるときには、フロー効率を重視した方がいい。そして、ソフトウェア開発の仕事の多くはそうである。だから、モブプログラミングはコストパフォーマンスの高い方法なのである。

図 1.4　リソース効率とフロー効率

1.3.2 キーパーソンに頼らない方法

同じ仕事をひとりではなくモブで行うメリットは、フロー効率だけではない。キーパーソンへの依存度が下がるというメリットもある。

ひとりで仕事をさせていると、彼らは自分がしている仕事への依存度を高めていってしまう。しかし、彼らがいなくなったらどうなるだろうか。組織は大きなダメージを受けてしまうのだ。

モビングで仕事を進めれば、チームに「ため」を作ることができる。複数の人々が仕事のしかたを覚えさせれば、そのなかの誰かが辞めても、組織が受けるダメージは小さくなる。

1.3.3 スキルアップ

キーパーソンへの依存度を下げることに加え、モブプログラミングにはチーム内の経験の浅い人々のスキルを早く引き上げる効果もある。

また、単独で仕事をさせていると、それぞれが独自の方法で仕事をするようになるが、モビングで仕事を進めると、チームメンバーは互いにそれぞれのアイデアを共有できる。こういう共有が蓄積されていけば、チーム全体の知識が広がり、全員がもっと有能になる。

1.3.4 クライアントに影響を及ぼすような欠陥が減る

最後になったが無視できないのは、数人で協力してひとつの仕事をさせると、仕事の品質が上がることだ。コードが書かれると同時に複数の人々がレビューするので、本番コードに欠陥が入り込みにくくなるのである。早い段階で欠陥を見つけて修復できると、サポートコール、バグフィックス、緊急パッチといったものが減るので、コストも下がる。言うまでもないが、本番コードの欠陥が減れば、クライアントの満足度も上がる。

2　モビングの準備

モブが初めて顔を揃えたときには、最初にモブプログラミングのコンセプトを説明しよう。これからの数節では、このときに話すべきことを説明する。

まず、モブとしてひとつの仕事をいっしょにすることの意義を論議しよう。

初めてのモビングセッションで行うべきことのおおよそのタイムテーブルは次のようになるだろう。

準備：お膳立て（30 分前後）

- モブという形でひとつの仕事に取り組むことの意義を説明する（10 分）
- モビングでの役割分担を説明する（8 分）
- モブが解決する問題の概要を説明する（10 分）
- タイピストの順番を決める（2 分）

本体：モビングインターバル（1 時間 30 分前後）

- 最初のタイピストにキーボードを渡す
- タイマーを 10 分にセットして開始する
- モブ全員が協力して問題の解決のために努力する
- 10 分経過したら、タイピストは作業を中止して次の人に交代する
- セッション終了の 20 分前まで、タイマーをセットし直して、インターバルを繰り返す

しめくくり：経験からの学習（20 分前後）

- それまでのセッションを振り返り、うまくいったこと、うまくいかなかったことを挙げ、次回の改善方法を議論する

2．1　モブという形でひとつの仕事に取り組むことの意義

ここで忘れずに言っておきたいことをまとめておこう。

- モブプログラミングとは、協力してひとつの問題を解決するために、1 台のコンピューターの前に数人が集まって共同作業をすることである。
- モビングを効果的なものにするためには、モブの一人ひとりのメンバーがフロー効率重視の考え方を確立していく必要がある。それは、課題の完成に向かって歩みが滞らないように、チームの一員として協力し合っていくという考え方である。
- フロー重視の考え方を確立するために必要なスキルを身に着けるためには練習が必要だが、決して不可能なことではない。世界中のチームがこれを成功させている。

モブプログラミングの具体的な方法について

モブプログラミングにもルールがある。キーボードの前に座っている人を「タイピスト」、タイピストと共同作業をするその他すべての人々を「その他のモブ」と呼称することにする

数人のグループが 1 台のコンピューターの前にひとつの問題を解決するために仕事をするとどういう感じになるのだろうか。下の写真のようにモビングする人々の 1 日を早送りで見せてくれる短い動画を見ると、イメージがつかみやすくなるだろう。

どういうやり取りをすべきかについては、文章の説明を読むよりも、ほかの人たちが実際にモビングしている様子を見た方がイメージをつかみやすい。ものごとには、誰か他人がやっているところを見た方がわかりやすいものがあるのだ。

動画を見たら、モビングでの役割分担を説明しよう。モビングには、タイピストとその他のモブのふたつの役割がある。

2．2　タイピストの仕事

タイピストとは、キーボードの前に座る人のことだ。ひとつのモブで同時にタイピストになるのはひとりだけである。タイピストは、プログラマーではない。むしろ、その他のモブがしてくれと言ったことを取り入れて実装するスマートアシスタントと考えた方がいい。

タイピストが効果的な仕事をするためには、次のことが求められる。

- その他のモブがしてくれと頼んだことを理解すること
- 要請の意味がはっきりしないときには、はっきりさせるための質問をすること
- してくれと頼まれたことをコードの形に実装すること

- その他のモブを信頼し、自分では通常試さないようなアプローチを躊躇せずに試す
- ショートカットキーやほかの人のツールの活用方法などの新しいことを学ぶこと

タイピストが回ってきたときには、その他のモブがしてくれと頼んでくることを理解するのが難しい場合がある。何を求められたのか理解できない場合もある。そういうときには、声を上げて、モブが求めるようなコードを書く方法がわかるまで質問をしなければならない。

有能なタイピストになるために大切な要素のひとつは信頼である。最良のタイピストは、自分が普通取るのとは異なるアプローチをその他のモブが選ぶことがあるのを認識し、そのアプローチの可能性を探り尽くすことをいとわないものである。生半可に抵抗するのではなく、要請されたことを実装してから、それを評価するようにしよう。

タイピストが説明を求めるのは、アイデアをコードに表現してからでなければならない。タイピストはひんぱんに交代するので、コードを理解していることが大切である。そうでなければ、その他のモブのひとりに戻ったときにチームに貢献できなくなってしまう。

しかし、タイピストをやっているときに、実装する価値があると思うアイデアが浮かんだときにはどうすればよいだろうか。その他のモブにアイデアを説明して実装に賛成してもらうか、ほかの人にタイピストを替わってもらって自分はその他のモブに戻らせてもらうかのどちらかになる。その他のモブに戻るときには、モビングインターバルはそこでリセットして、キーボードを渡された人が新しいタイピストになる。

最後になったが、タイピストを務めているときは、自分のツールの異なる使い方をほかのメンバーから学ぶすばらしいチャンスだ。ほかのメンバーが自分とは違うやり方や、作業ペースが上がる新しいショートカットキーを教えてくれたときには、受け入れるようにしよう。こういった小さな改善によって自分の仕事がいかに早くなり、能力がいかに上がっていくかは驚くほどだろう。

2．3　その他のモブの仕事

モブのなかでタイピストではないその他大勢の一部になっているときも、すべきことはたくさんある。

- 問題解決につながる次の論理的ステップを見つけるために力になること
- 理解できるまで質問をすること

- モブ全体の理解の水準を上げるために貢献すること
- 眼の前の問題に集中すること
- ほかのメンバーの意見を聞くこと
- 必要な情報を予測すること
- システムのなかの改善すべき部分を探すこと

モブのかなでタイピスト以外になるということは、問題解決チームの一員になるということだ。目の前の問題の解決につながる次の論理的ステップを見つけるための力にならなければならない。これはほかのメンバーと協力して仕事を進め、自分が理解できるまで質問をしていくことによって実現できる。

問題にはかならずさまざまな側面とアプローチがあるが、次の論理的ステップは何かについての意見を一致させるために努力しなければならない。その他のモブのひとりとしては、全員が理解できるようなレベルの話をして、意見の一致のために積極的に動くことが求められる。

たとえば、一部をメソッドに切り分ければコードを単純化できるときには、タイピストにそうするよう求めればいい。タイピストがぽかんとしているようなら、具体的に「114 行から 127 行を選択してから[Ctrl-Alt-M]を押してその部分をメソッドにしよう」と指示する。

その他のモブの一員になっているときには、コミュニケーションがすべてだ。自分の考えを明快に説明し、周囲に理解してもらわなければならない。

私たちは、効果的なコミュニケーションのために必要なのは、何を言うべきかに集中することだと思いがちだ。もちろん、それも大切だが、それはコミュニケーションスキルの一部に過ぎない。本物のコミュニケーションで必要とされるのは、ほかの人の意見をよく聞くことだ。次に自分のアイデアがなかなか伝わらないと思ったときには、自分がちゃんと話を聞いているかどうかをチェックしよう。聞けていないことがよくある。それが問題の半分だ。

他人の話をよく聞こう。モブに属する全員、特に話を聞いてもらえていない人々に十分に気を配るようにしよう。

モブはときどき脇道にそれて迷走することがある。そうなっていないかどうかに注意しよう。迷走が起きているときには、メンバーに注意喚起し、本筋に戻るための努力をしよう。目の前の問題に直接関係のないアイデアはしばらく脇に置こう。

必要な情報を予測し、積極的に探そう。必要になる前にドキュメントを探して関連箇所を強調表示しておいてくれる人がいるといかに役に立つかは驚くほどである。これをするチャンスに気が付いたら、実施するようにしよう。

そして、作業やコードのなかの改良できる部分を探そう。繰り返されるタスク、プロセスは、自動化すべきだ。ほかのメンバーが知らないショートカットキーやエディターの操作テクニックがあれば教えよう。怪しげなコードや不必要に複雑なところを見つけたり、ものごとを単純化する方法を探したりするのも、その他のモブとしての仕事の一部だ。

2.3.1 モビングで解決しようとしている問題の概要を紹介する

モブたちにふたつの役割が果たすべきことを理解してもらったら、解決しようとしている問題を紹介しよう。モブとして、問題にどう取り組んでいくかについてのおおよそのプランは一致していなければならない。ホワイトボードに問題解決のためのさまざまなアプローチを書いて全員でブレインストーミングしよう。ただし、そういった議論はせいぜい 15 分程度で終わらせるべきだ。意見が一致したら。ホワイトボードにおおよその手順を書き出し、実装の具体的な細かい問題はモビングインターバルで考えるようにする。

2.3.2 タイピストの順番を決める

問題について説明したら、タイピストになる順番を決めなければならない。すべてのメンバーにタイピストになるチャンスを与える。これを徹底するために、モブのメンバー全員がタイピストを務めてからでなければ、2 度目のタイピストが回らないようにしよう。

タイピストの順番はどう決めてもかまわないが、決めた順番は全員が見られるように書き出してどこかに掲示しよう。タイピストを是非やりたいというメンバーがいれば、リストの先頭に置いていい。誰もやりたがらない場合には、無作為に選んでいって結果を書き出す。誰が最初にタイピストを務めるか、どういう順序になるかに大きな意味はない。順序が明確で、全員に同じようにタイピストになる機会が回ってくるならそれでよい。

ときどきモブのなかに、タイピストになりたがらず、その他のモブは喜んで務めるというメンバーが出てくることがある。見ているだけなら安心して参加できるというなら、それは受け入れていい。安心は大切である。とは言え、全員にタイピストとして参加することを奨励することも大切だ。他人が問題を解決するところを見ているのと自分の手を汚すのとでは違いがある。モブの一員として実際にキーボードを叩くと、問題に深く入り込みやすくなるし、問題解決能力が動き出すようになる。

ときどき、その分野のコーディングは自分の専門外だとか、全体の作業スピードを遅らせるのではないかと言って、タイピストになるのをためらうメンバーが出てくることがある。そういうときには、次のように説明しよう。まず第 1 に、エキスパートではない人にとって、タイピストは最良のポジションだ。タイピストという役割は、定義上、その他のモブがしてくれといったことをすればいい。エキスパートでならなければならないのは、その

他のモブの方だ。第 2 に、知ることとすることには違いがある。目標は、することができるというレベルに全員を引き上げることだ。タイピストを務めることは、「する」ことができるようになるための近道である。

2．4　モビングインターバル

最初にタイピストを務める人を選び、最初のモビングインターバルを始められる状態になった。しかし、インターバルとセッションはどう違うのだろうか。まずそこから説明しておこう。

モビングセッションとは、人々がひとつのグループとしてモビングする継続的な時間のことを言う（通常、数時間に及ぶ）。それに対し、モビングインターバルは、ひとりがタイピストを務めている短い時間のことを指す（通常数分）。モビングセッションは、準備、一連のモビングインターバル、最後のまとめ作業から構成される。モビングセッションの時間の大半は、モビングインターバルに費やされる。では、モビングインターバルの具体的な内容をもう少し詳しく見てみよう。

2.4.1 最初のインターバルを開始して 10 分間モビングする

モビングインターバルは、10 分間のタイマーをセットして始まる。10 分にすると、モブのメンバー全員に数回ずつタイピストの番が回ってくるのでモブセッションがうまく回る。

ここで早速難問が現れる。どこから始めたらよいのだろうか。問題にどういう角度からアプローチするから人によって異なる。たとえば、トップダウン方式とボトムアップ方式とでメンバーの好みが割れる場合がある。私は TDD（テスト駆動開発）を実践しているので、最初に考えるのは「まずどういうテストを書いたらよいか」だが、TDD を使っていない人なら「まずどこを書いたらよいか」だろう。

経験の浅いモブは、最初のインターバルでは、あまりコードを書けないことが多い。インターバルに入っても、すり合わせが必要になることが多いのである。これはまずいことではないが、できる限りコードを書く方向に持っていきたいところだ。コーディングに進むと、会話の内容が理論的なものから実践的なものに変わっていく。問題へのアプローチのしかたがまちまちなら、最初はもっとも単純なものを選ぼう。それで回るなら、通常それは十分いいアプローチだ。しかし、それでは回っていかないようなら、コードを書き始めてすぐに欠陥が明らかになる。同じように単純なアイデアが複数あるなら、どれかひとつに絞り込もう。問題の解き方はたくさんあるということは、モブプログラミングから学べることのひとつだ。

2.4.2 最初のインターバルの最後でタイピストを交代する

最初のインターバルでは、タイマーに注意しよう。タイマーが切れたら、インターバルをすぐに終わらせ、タイプストを次の人に交代させなければならない。タイピストは、その時点でしていることを「終わらせたい」と思うものだが、その気持ちはぐっと抑えるようにしよう。新しいタイピストとの交代はすばやく終わらせたい。

交代したら、タイマーを再び 10 分にセットし直して、カウントダウンを開始する。また、この機会を利用して、バージョン管理システムに作業内容をコミットしておくといい。

10 分間のインターバルを初めて経験して感じるのは、10 分はコードを書くためには本当に短い時間だということと、それでも少しは仕事を前進させられるということだ。短時間のインターバルを守っていくと、全員の集中を保つために役立つ。

2.4.3 その後のインターバル

最初のインターバルを経験すると、モビングセッションのその後のインターバルのパターンができる。グループ一丸となって問題に取り組み、10 分のインターバルでタイピストを交代しよう。

このなかには、モブの全員が次に何をすべきかについて一致しており、それをコード化するだけだという、スパートをかける時期が含まれるだろう。また、ホワイトボードでアイデアを説明しなければならない時期も含まれるだろう。インターバル全体をホワイトボードの前で使ってタイピストの仕事がなくなってしまった場合には、コーディングを再開したときに、次のタイピストのインターバルをそのタイピストに回すとよい。

インターバルごとに、モブのなかでそのときの自分の役割を意識しよう。タイピストの順番でなければ、モブのなかでのそのときの自分の役割を意識しよう。タイピストの順番でなければ、キーボードに触れてはならない。初めてモブに参加する人にとっては非常につらいことで、コードを書いて説明したいという不平不満が出る。しかし時間とともに説明することと他人の話を聞くことがうまくなってくると、こういった不満は減っていく。

モビングセッションの冒頭でホワイトボードに書いたおおよその手順に絶えず立ち返るようにしよう。全体的なアプローチを変えることにしたときには、それに合わせておおよその手順も更新しなければならない。ボードの一部は ToDo リストのために取っておこう。目の前の問題からそれてしまっても検討すべき大切なアイデアやタスクは ToDo リストに書き留めるようにすべきだ。

モブの参加者が 3 人で、インターバルが 10 分なら、最初のセッションでタイピストの順番が回ってくるのは高々3 回である。時計から目を離すと、モビングインターバルではいかに速く時間が過ぎていくかに驚くだろう。しかし、セッション終了の 20 分前には

インターバルを終了するのを忘れないようにしよう。まとめと振り返りのための時間を十分残しておく必要がある。

２．５　経験からの学習

セッションの最後の 20 分は、モビングを締めくくるためのレトロスペクティブ（反省会）に充てる。ここでは、モビングに参加したメンバーが感じたことを述べ合い、次回に軌道修正すべきことを提案する。

3. モビングの普及について

現状日本で多く用いられているSES(System Engineering Service)形態での雇用形態の場合、非常にモビングを行いにくいと考えられる。

インターネットでのモビング活用例としては、
- 自社開発のプロダクトがある
- 技術重視（ないしは技術中心）
- 比較的ベンチャー企業気質がすでにある

この3つが備わっている企業での活用が多い。

今回説明を割愛したが、モブプログラミングを行う上での一番の問題点は、自社でも派遣先でも1つの作業を複数人で行うことに対してのコスト意識になる。
自社であれば上司や役員など、技術を中心として考えている企業であれば、比較的導入に関しての説得は容易だと思われる。

ただ日本独特の派遣先では非常に難しいのでは？と私は考える。
仮に一人60万の人月派遣で5人のチームだとしよう。このチームが一つの画面を見ながら一つの作業を行っているという状況を派遣先が見て起こる批判は
- 5人それぞれで作業をすれば5つの作業ができるのでは？
- タイプをしている人以外は何をしているの？

などの問題があり、一つ一つはねのけてまで導入することは非常に厳しいと思う。

この状況を許容できる派遣先があるのであれば、問題なくモビングを導入できると思う。
おそらく営業的要素を考えても現状99．99％批判されることになると思う。

4 考察

ここまでモビングについて話してきたが、現状自社では導入できていない。
理由としては開発者が私一人だからである。

複数人（3 人以上）の人間がいる開発現場で、ホワイトボードや作業スペース、作業環境の構築も考えると現状導入できるのは
「開発者が複数いるベンチャー企業」一択になるのではないかと思われる。

従業員数は多いけれど、ほとんどが派遣先にいるというケースが多い日本のシステム開発会社では非常に導入しづらいモビング。

ただ日本には古来より「三人寄れば文殊の知恵」という、特別頭の良い人ではなくても3人集まって相談すればよい知恵が浮かぶということわざがある。
私自身は今回の学びからこの「三人寄れば文殊の知恵」をプログラムに応用したものがモブプログラミングだと解釈している。

モブプログラミングは単純なリソースの管理だけでは解決できない複雑な問題が多々ある開発現場では非常に有益な開発方法だと思う。

偶然ではあるが、十数年前モビングやペアプログラミングなどの単語など意識しなかったとき、現場のチームで同じような試みが行われたことがある。
それはこうしようと思って行われた能動的なものではなく、チームのメンバーの息があった結果と元請け会社が完全に現場を放棄していて権限が現場にあったという状況が生み出した偶然の産物だった。

しかしその時の効率は非常に良かったと記憶している。

その時の思い出が今回この本を手に取り学ぶ機会になったことをもって終了とさせていただく。

参考文献

(1) マーク・パール　著、長尾高弘　訳、及部敬雄　解説、モブプログラミング・ベストプラクティス ソフトウェアの品質と生産性をチームで高める、日経 BP、2019 年 2 月。

デザイン思考等の考察　報告書

発行日	2022 年 1 月 18 日　第 1 版　発行
発行者	ITC大阪城
作成者	ITC大阪城　テーマ研究調査活動　ワーキンググループ 新　保　康　夫 脇　阪　公　昭 山　中　美　智　子 岡　田　誠　司

ITC大阪城