「作らない開発」とアジャイル

No-making Development and Agile

桝田秀和

要　約　「作らない開発」は「再利用」「オーバーエンジニアリング防止」「自動化」を指すものである．ソフトウエア開発において「作らない開発」を推進することは生産性を向上させビジネス企画やその価値向上にリソースを集中させるために重要である．「モダン開発」と呼ばれるクラウド技術や開発ツールなど最新技術を取り入れることで生産性と品質の同時向上を達成できる．

モダン開発の主要要素として「アジャイルUX」という仮説検証型の開発方法論がある．反復的に価値を探索するスタイルを取り入れ，無駄なソフトウエアを作らないことが「作らない開発」において重要である．

Abstract　No-making Development refers to reuse, prevention of over-engineering, and automation. Promoting No-making Development in software development is important to improve productivity and focus resources on business planning and value enhancement. By incorporating the latest technologies such as cloud technology and development tools, collectively known as modern development, it is possible to achieve simultaneous improvement in productivity and quality.

One of the key elements of modern development is a hypothesis verification-type development methodology called Agile UX. It is important in No-making Development to incorporate a style that explores value iteratively and does not create unnecessary software.

1. はじめに

バブル崩壊以降，ICTが社会に浸透する中で，日本の国際競争力も低下している[1]．日本企業はこれまで，デフレ時代に対応するためにコストカットを中心とした事業運営を行ってきた．結果，IT投資が減少しIT国際競争力が低位に甘んじている状況である[2]．現在行われている経済政策の転換にあわせて，ITの能力を最大限に活用した事業運営を行い，高付加価値化を推進することが求められている．日本の生産年齢人口の減少に対処するためにも，あらゆる企業はITを活用する能力を組織的に身につけなければならない．

あわせて，2022年に登場した生成AI（Generative AI）技術が我々の働き方を大きく変えようとしている．個々人の能力を高める技術は，能力の格差を拡大する．ITを適切に取り入れ，活用できる能力を従業員に身につけさせることが重要である．

一方で，経済産業省がまとめたDXレポート[3]にもある通り，多くの日本企業は既存システム，いわゆるレガシーシステムの保守に多くのコストを費やしている．そのコストを低減しつつ，新しい技術分野に投資を行うためには，企業がシステムの開発運用を業務委託しているシステムインテグレーター（SIer）との協調が不可欠である．

本稿では，2章でソフトウエア開発における「作らない開発」の適用が重要であること，3

章で歴史的な経緯を紐解きながら「モダン開発」について述べた後，4章では「アジャイル」が特に重要な概念であることを述べる．5章では，BIPROGY株式会社（以下，BIPROGY）が提供しているモダン開発のための基盤と支援サービス「AlesInfiny（アレスインフィニィ）[*1]」について紹介し，AlesInfinyを適用することで「作らない開発」を実施できる組織に変革する道筋を説明する．6章では，生成AI技術が「作らない開発」に与える影響の現状と展望を述べる．

2. 「作らない開発」とは何か

ITシステムは，その機能の多くを「ソフトウエア」にて実現する．ハードウエアと異なり，ソフトウエアは物理的な制約が少なく，再利用が容易であるという特徴を持つ．あわせて，物理的に目に見えたり触れたりすることができないため，本当に欲しいものは何かが分かりづらいという特徴も持つ．

「作らない開発」とは，ITシステムを開発保守する際に，ソフトウエアの特徴を考慮に入れながら，生産性と品質，自身の再利用性を高めるための以下のような活動を指す．

- 本当に利用者に喜ばれる機能しか「作らない開発」
- 開発保守を行う上で，他の本番稼働がなされた部品を再利用することで，作るべき範囲を業務に必要な最低限のものとする「作らない開発」
- 様々な開発・保守の活動を自動化し，コンピューターに移管することで人手を減らす「作らない開発」

上から順に「オーバーエンジニアリング防止」「再利用」「自動化」と呼ばれる．これを踏まえ，競争優位性を求めない機能は他社のサービス（SaaS）を利用し，競争優位性を確保するための企画・開発・運用機能はできる限り自社に保有し，市場の変化に追従できる体制を作り出すことが肝要である．このようなITシステムの新規開発や移行を行う際には，最新のITシステムに関する素養（目利き力）が不可欠である．ITシステムの構築経験を豊富に持つベンダー，すなわちSIerが，最新のITシステムに関する知見を企業に提供し，ITシステムの高度化による恩恵を受けられるようにすべきである．

2000年代からの20年間で，ITシステムを取り巻く環境は大きく変化し，開発の現場においても生産性に大きな格差が生じている．過去のITシステムの現場は労働集約的であったが，現在では単純労働の多くはコンピューターで自動化できるようになっている．しかし，日本においては現場作業（コーディング作業）を外部委託することが大半であり，コストカットの要請からオフショア化も進んだため，労働集約的な構造・作業マニュアルが改定されないまま現在に至っている．ゆえに，生産性の向上の恩恵を受けられる現場が少ないのが現状である．

ビジネスとして本来あるべき姿を目指しても，生産性を高めない限り対応に時間がかかり，市場環境の変化に追従できない．BIPROGYは，ビジネスの価値向上にリソースを集中させ，企業の競争力を向上させるため，現場の生産性を高める「作らない開発」を推進している．

3. 「モダン開発」とは何か

本章では，2000年代からのITシステムの開発運用に関する技術的な進化について，その歴史的経緯を紐解きながら「モダン開発」とは何かを図1の概念図に沿って説明する．

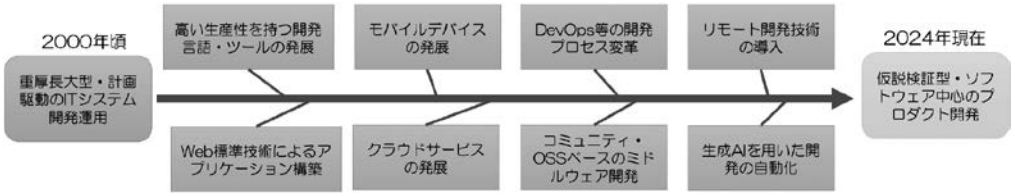

図1　IT システム開発運用の技術発展

3.1　高い生産性を持つ開発言語・ツールの発展

2000 年代に入ると，インターネット，特に Web が一般に普及する．これにより，Web を通じたサービスを提供し，収益化する企業が登場した．HTTP や HTML といった Web 技術は現在も基幹技術として IT システムを支えている．これらを活用した Web サービス，後に SaaS と呼ばれる事業形態が生まれてきた．

Web サービスを提供するスタートアップ企業は，Ruby on Rails などの軽量 Web フレームワークを好んで使うようになった．当時一般的だったフレームワークでは長大な設定ファイルを記述しカスタマイズしていた．軽量 Web フレームワークの登場により，一定の規約に基づいて記述することで，自動的に設定がなされるようになり，業務のロジックのみに集中できるようになった．後に登場するフレームワークも，軽量 Web フレームワークのノウハウを取り入れることになった．

3.2　Web 標準技術によるアプリケーション構築

多くの PC にインストールされているブラウザで動作する Web アプリケーションが事実上の標準となり，HTML や JavaScript といった言語が発展することになった．HTML5 は，これまでマルチメディアの表示を主に行う仕様であった HTML を大きく拡張し，プログラムの動作基盤としての Web を標準化するものになった．特に，Web アプリケーションをデスクトップアプリケーションと同様に滑らかに動作させる技術である Ajax の普及は，後に REST API という外部サービス連携のための仕様の標準化に繋がり，ブラウザを中心に様々な SaaS を連携動作させ，ブラウザで統合させて機能を提供する，という仕組みに繋がっていく．

3.3　クラウドサービスの発展

2010 年代には，Amazon Web Services などのクラウドサービスが本格的に利用されるようになり，災害対策やバックアップの取得が容易になることで広まり，サーバーの所有から利用という考え方が浸透していくことになった．また，Microsoft は競合サービスである Azure を提供し，PaaS（Platform as a Service）を中心にサービスを提供した．クラウドサービスの利用者である開発者はソフトウエアだけを考えれば良く，インフラの設計や運用についてはクラウドベンダーが責任を持つ，という考え方は，一般にマネージドサービスと呼ばれる．システム開発運用を行う場合は，共同利用・再利用部品であるマネージドサービスを活用して素早く自らの事業価値を提供することが重要になった．

3.4　モバイルデバイスの発展

2010 年代はスマートフォンの普及の時代でもあった．スマートフォンやタブレットが浸透

するにつれ，利用者は操作性の高いアプリケーションを求めるようになった．モバイルアプリケーションの開発技術が独自に発達していくとともに，Web アプリケーションもモバイルに適合した形に進化していった．React を代表とする Single Page Application（SPA）は，Web アプリケーションを 1 ページで構成し，画面変化を全て JavaScript で記述することで操作性を向上させることができた．SPA を筆頭としたフロントエンド技術は独自の発展を遂げ，Web アプリケーションの開発は，フロントエンドエンジニアとバックエンドエンジニアという個別の役割で遂行されることが多くなった．さらに，B to C，C to C などの事業領域においては，顧客接点がスマートフォンに移動するのに伴い，モバイルファーストという考えが広がり，モバイル技術に特化したモバイルエンジニアという役割も生み出した．

クラウドサービスなどのバックエンド技術については仮想化技術を用いた PaaS が発展し，標準化されたコンテナという技術が普及した．これにより，開発環境とテスト環境，本番環境との間の環境差異がなくなり，アプリケーションの移植性が向上した．PaaS 基盤上で動作するコンテナアプリケーションは，稼働負荷の増減に追従してサーバーは生成・削除されるものとなり，インフラ担当の役割も変化していった．それを端的に表す言葉が DevOps である．

3. 5　DevOps 等の開発プロセス変革

DevOps の登場により，PaaS やコンテナを活用して，従来はインフラエンジニアの担当だった作業を Web アプリケーションの開発者自身が行うようになり，開発リードタイムを削減することができるようになった．インフラエンジニアは，システム開発運用の手作業（トイル）を削減し，自動化しリリースサイクルを高める SRE[*2] という新たな役割を担うようになった．

DevOps の中心にあるのは，Git 等のソースコードリポジトリであり，リポジトリを集中管理しアプリケーションをビルドするためのビルドパイプラインの仕組みであった．これらを総称して継続的インテグレーション（Continuous Integration）・継続的デリバリー（Continuous Delivery）と呼び，CI/CD と略される．開発者の変更活動はブランチと呼ばれる単位で管理され，レビュー後に統合され，統合直後に自動的に静的解析とテストが行われ，品質のチェックがなされる．継続的なインテグレーションとテストは統合時のリスクを大きく下げ，デグレードを防止するセーフティネットの役割を担っている．素早い変更を安全に，既存システムに統合できる仕組みの整備は，開発全体をより活動的にし，事業価値を高めることに繋がる．

本稿では，最新の IT システムが持つ，高度な生産性と品質を両立させるための機能群を活用することを，開発現場の現代化という意味で「モダン開発」と定義する．モダン開発を行うことで，「作らない開発」の基礎的な能力を獲得することができ，企業の競争力向上のために求められる IT システムを正しく構築・調達することができるようになる．DevOps 等を活用して自動化による生産性や品質の向上を図る開発プロセスは，モダン開発の典型例である．

DevOps の発展が進み，セキュリティを短い開発サイクルの中で確実に取り入れる仕組みである DevSecOps という概念も登場した．CI/CD パイプラインにセキュリティスキャンを組み込むことにより，統合テスト時にはじめて明らかになる脆弱性を開発時点で検出できるようになった．セキュリティテスト工程を前倒しする「シフトレフト」を推進することで，セキュリティ検査のために開発期間に待ちが生じることを防止できる．脆弱性を攻撃されることで事業価値を毀損するリスクを低減しながら，素早い開発サイクルを維持することが，ツールの支援によって実現できるようになった．

3.6　コミュニティ・OSS ベースのミドルウエア開発

前節で述べた諸活動を支える各種ツール/ミドルウエア/PaaS の多くは，オープンソースコミュニティにより生まれてきた．GitHub などのコミュニティを支える基盤としてリポジトリやコンテナを前提とした技術が採用されている．これらの技術自体もオープンソースコミュニティにより生まれ，改善されていく．オープンソースコミュニティは，不特定多数の世界中の人間が顔を合わせることなく文字ベースでコミュニケーションを行い，数百万行にわたるコードを開発して保守するというコラボレーションができることを示した．我々のようなビジネスに関わるアプリケーション開発者も，このようなコラボレーションツールや文化を導入することで自由な働き方でより効率的に活動できるはずである．

3.7　リモート開発技術の導入

コラボレーションツールの効果を現実に示したのが，2020 年に発生した COVID-19 の流行である．世界中の企業がリモートワークに強制的にシフトする中で，生産性を維持するため様々なコラボレーションツールを用いてリモートでの開発に挑戦した．オンラインのホワイトボードツールやプロジェクト管理ツールの利用，チャットツールによる開発運用，そしてオンラインでの共同作業により地方や海外のチームとの協業を行うようになった．クラウド環境での仕事環境が前提となるにつれ，ゼロトラストセキュリティのような高度なセキュリティ基盤の必要性が認識され，普及し始めている．

3.8　生成 AI を用いた開発の自動化

2022 年末に登場した ChatGPT などの生成 AI 技術により，大きな時代の変革がまさに始まろうとしている．おそらく，Web の登場と同等のインパクトをもたらすこの技術は，コンピューターがあたかも知能を持っているかのように振る舞うことを示し，実際，様々な業務を人間の代わりにこなすことができている．利用価格やプライバシーの扱いなどの問題もあり B to C などのアプリケーションへの組み込みは始まったばかりの状況であるが，先進的な開発者は，自らのシステム開発運用業務にこれを取り入れ，業務の副操縦士（Copilot）として活用している．これらの基盤技術である大規模言語モデルはいまだ発展途上であり，将来的には対話や様々な問題の解決を人間と同じように行うことが期待されている．

生成 AI を用いたシステム開発運用は，「作らない開発」における自動化に新しい可能性を与える．一方で，AI が理解できる大量の文脈情報（コンテキスト）を事前に与えなければならない．このため，通常の開発とは異なる「AI ネイティブな」開発方法論が求められる．具体的には，様々な用語やデータ，設計情報を，生成 AI の入力になるようなテキストベースの，質の高いものとして与えることが重要である．また，生成 AI が適切な回答を返すためには，曖昧性なく自らの意図を伝える適切な「プロンプト」を与えなければならない．

3.9　仮説検証サイクルを回すアジャイルとアジャイル UX

本章では，歴史的な流れを追いながら様々な要素技術について紹介してきた．これらの技術は，いずれもコンピューターの力を活用して自動化や再利用を行う「作らない開発」のための技術である．これらの技術は，当然，コストカットにも活用することができるが，多くの先進的な企業では競争力の確保のために活用している．すなわち「仮説検証を素早く回し，市場に

自らの考えを問い，その結果を事実として捉え，方向転換を行いながら事業を成長させる」という仮説検証サイクルを回すために利用している．

仮説検証サイクルを事業全体で行うことを，スタートアップ事業者の文脈では「リーン・スタートアップ」と呼ぶが，一般にはこれらの活動を総称して「アジャイル」と呼ぶ．その際に，利用者の視点を正しく捉え，真に喜ばれる機能のみを作っていくための方法論として，ユーザー体験（User Experience, UX）を重視することが求められる．BIPROGY は，アジャイルかつユーザー中心の考え方を推進するための活動を総称して「アジャイル UX」と呼び，開発現場での実践に取り組んでいる．

4. 作らない開発の中心要素「アジャイル UX」

作らない開発のうち，本当に利用者に喜ばれる機能しか「作らない開発」，すなわちオーバーエンジニアリングを防止することが最も重要である．なぜなら，使われない機能であっても作ることで，サービスの稼働，変更の取り込みや保守にコストがかかるためである．機能を絞り込み，絞り込んだ機能を磨き上げることが，利用者や従業員の満足度を向上し，システムの投資対効果を高める．

4.1 アジャイル実践の現状

アジャイルは，よりよい開発のあり方を目指す開発者たちが，ウォーターフォールと異なるパラダイムの開発の姿を「アジャイルソフトウエア開発宣言」として表したことから始まった．「個人と対話を繰り返すことで，動くソフトウエアを素早く作り，見せることで顧客と協調することを促し，長期の固定的な計画ではなく，変化に対応できる能力を獲得することが重要である」というコンセプトは，VUCA[*3] と呼ばれる不確実・予測不可能な現代における経営のあり方にマッチしたため，世界的なムーブメントになった．DX 白書 2023 によると，IT 国際競争力がトップクラスである米国では，IT 部門の 8 割以上，他部門でも 6 割以上がアジャイルを実践しているとされる．一方，日本では IT 部門であっても半数以下の組織しかアジャイルを実践できていない[4]．

4.2 アジャイルのメリット

アジャイルを組織に取り入れることのメリットを述べる．まず，2 割の少数が 8 割の貢献を生み出している，という「パレートの法則」がある．2 割の主要な機能を素早く作ることで，全体的な価値を素早く獲得できると考えると「小さく作って顧客を巻き込みながら価値を確認する」という進め方が合理的である．建築物のようなハードウエアとは異なり，ソフトウエアは作った後に変更することができる．自動テストのような技術を適用することで，ソフトウエアの変更にかかるコストを低減できる．そのため，1 週間から数週間程度の短期間で全ての開発工程を実行し，これを繰り返すことが効果的である．

アジャイルではない従来の開発スタイルでは，顧客が要件や仕様を SIer に伝えた後，受入試験まで実物をほとんど確認することのないまま待ち状態に入る．アジャイル実践のための有名なルールセットである「スクラム」では，一定の「スプリント」と呼ばれる開発単位の中で，「スプリントレビュー」と呼ばれる顧客巻き込み型の発表会を開催する．顧客は実際に動作するシステムを触りながら自らが伝えた要求を確認できるとともに，ビジネス状況の変化に応じ

た新たな要求を伝えることができ，チームは変更を歓迎する（図 2）．

また，アジャイルを組織に取り入れることで「継続的改善」を実践できるというメリットもある．スクラムでは「スプリントレトロスペクティブ」と呼称される活動の振り返りと改善の場が設定されている．一旦立ち止まって自らの活動を振り返り，改善点をあげて，開発プロセスの変更を行う．次の開発工程の中で効果を測定し，改善を繰り返すことで複利的にチームの能力を高めることができる．

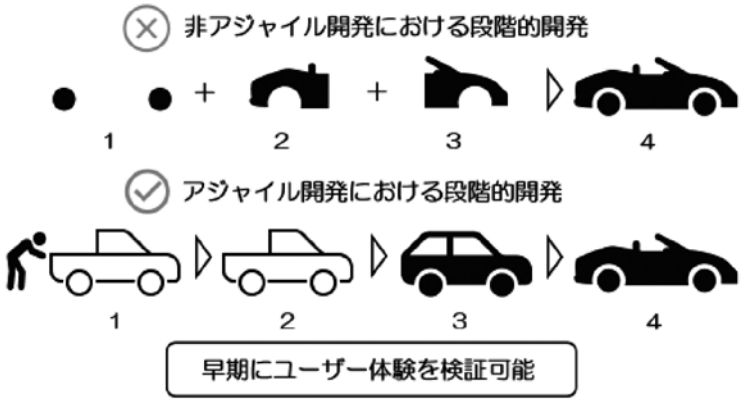

図 2　従来型開発とアジャイル開発の違い

4. 3　ユーザー体験（UX）を最大化するデザイン思考

利用者に使いやすく愛されるアプリケーション（ソフトウエア）を作成することで，事業目標を効果的に達成することができる．一般利用者向けのアプリケーションはもちろん，SoR（System of Record）と呼ばれるシステムにおいても，最新のアプリケーション体験に慣れた従業員の満足度を高めることが重要である．ユーザー体験（UX）とは，製品やサービスを通してユーザーが感じる使いやすさ，感動，印象といった体験すべてを指す．UX を最大化するためのデザイン方法論が存在し，「デザイン思考」と呼ばれる．

デザイン思考においては，単なる機能としてのデザインを作成することを目指すのではなく，事業目標に対して意義ある利用者の「行動の変化」を目指す．機能の完成だけでなく，そこからの成果を計測するために，利用者による検証を行う．検証結果を分析・学習し，さらなるデザインの改善を積み重ねることを反復する．これを効率的に行うためにツールセットやプロセスを活用する．もっとも重要なことは実利用者たる人間の観察であり，我々自身の思い込みをできる限り排除しなければならない．

デザインを作り上げるにあたって，利用者の観察やインタビューを通して解決すべき課題を抽出するのだが，以下の課題があり難しい．

- 利用者は自らが欲しいものを言葉にすることができない
- 利用者が課題だと思っていることが，真の課題ではない
- 具体的な「もの」を通して事後的に自らが欲しいものを想起する

これらの課題に対処するため，検証（実験）と結果の評価，そこからのアクションを繰り返す．これはアジャイルと全く同一の考え方である．アジャイルに，UX を最大化するデザイン思考を取り入れた活動が，アジャイル UX である．

4. 4　アジャイル UX のチーム構成

小さなチームに UX デザイナーと開発者を含め，共創のセッションを行い利用者の課題をともに認識する．学びを促す環境や会議体を設定し，自らの行動や行動の根拠となる考え方がアジャイルの精神を伴っているかをその場で指摘できる「コーチ」を用意する．一つのプロダクトバックログと呼ばれる活動リストを共有し，優先順位を共同で設定し，価値の高い作業から着手し，分担して開発と検証・評価を行う．このようなチームを構成し，パレートの法則の知恵を活かすために「何が主要な価値か」「何を作るべきでないか」を精査し素早く価値を実現することができれば，活動は成功事例となり，成功事例を梃子にして組織全体に競争力のある進め方や組織のあり方を展開することができるだろう（図 3）．

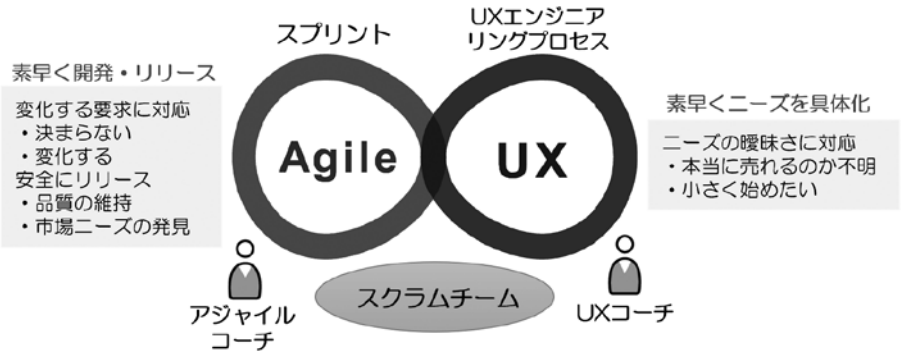

図 3　アジャイル UX の実現モデル

しかし，開発チームが持つべき前提条件がある．短い期間でアイデアやデザインを着実にアプリケーションとして実装できる確たる能力が不可欠である．セキュリティなどの品質も疎かにすることはできない．着実に高品質のものづくりを実現するためには，既に本稼働している他プロジェクトで品質が担保された部品やノウハウを再利用し「作らない開発」を実践することが求められる．

5.　モダン開発を支える AlesInfiny

BIPROGY は社会課題を解決する企業として複数の自社サービスを開発・運用している．この経験を共通化・抽象化し，現代のアプリケーションやシステムに求められる品質を確保できる複数の共通利用サービスを提供している．

AlesInfiny は，BIPROGY が提供するアプリケーション開発・運用領域における最新のサービスラインナップである．モダン開発を支えるための DevSecOps 統合基盤と伴走型サービスを提供することで，ビジネスに重点をおいたアプリケーション開発に専念でき，ビジネスの最新化と敏捷性の獲得を実現できる．AlesInfiny が解決できる課題領域とソリューションを図 4 に示す．

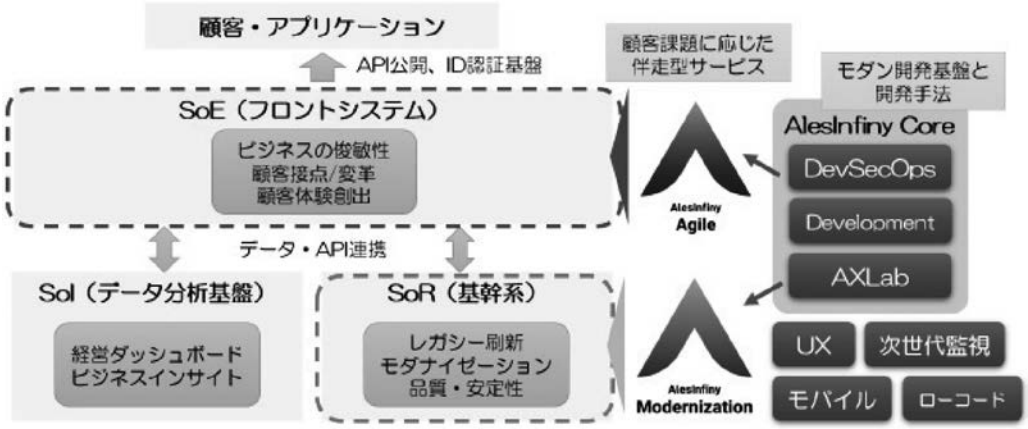

図 4　AlesInfiny が解決する課題領域

AlesInfiny Modernization は，企業の DX 実現に向けて最新技術でシステムをモダン化するためのソリューションである．AlesInfiny Agile は，新規ビジネスの事業創出を支えるシステムを仮説検証型で作成するためのソリューションである．いずれも，モダン開発基盤と開発方法論を適用し，組織が新しい開発スタイルを実践できるようにするものである．

モダン開発基盤と開発手法をあわせて，AlesInfiny Core と呼ぶ．クラウドネイティブなアプリケーション開発フレームワーク，クラウド上のアプリケーション開発/実行環境，高セキュリティ運用基盤（DevSecOps 基盤）と，これらを適用し，活用するためのコンシェルジュサービスを含む．

新しい開発スタイルを組織に取り入れるためには，単なるツールや方法論が存在するだけでは不十分である．人材育成や教育を行い，チームに伴走し寄り添いながら足りないスキルやマインドセットを指摘し，成長を促していかなければならない．高度な技術的課題に対しても気軽に質問できて，フォローしてくれるプロフェッショナルサービスも欠かせない．AlesInfiny は，これら組織浸透のためのサービスを提供している．

5. 1　AlesInfiny が提供するサービスメニュー

既存の情報システムが持つ課題，新規ビジネスの事業創出に関する課題は複雑に絡み合っており，解決可能な形に分解しなければならない．相談窓口となるコンシェルジュが事前検討段階から本番稼働まで相談に応じ，多様な技術的背景に基づき AlesInfiny が提供できるサービスメニューを適用し課題を解決する．

AlesInfiny が提供するサービスメニューには，「モダナイゼーション検討」「アジャイル」「モダン開発基盤」「オブザーバビリティ」「内製化」などがある．

「モダナイゼーション検討」は，最新技術に適応するための ICT サービスを検討する際に，どういう方向性で進めていけばよいのか分からないという課題に対して，最新技術の知見を持つ経験者がアセスメントを行い，どのようにシステムをモダナイゼーションすれば良いのか，クラウド化をどういうステップで進めるべきかを支援するサービスである．

「アジャイル」は，ウォーターフォールのような長期間に渡る計画駆動の開発プロセスでは変化する要件に追従できないという課題に対して，経験を積んだコーチからの指導によりチー

ムにアジャイルな開発スタイルをレクチャーする支援サービスである．アジャイルコーチの指導により，フィードバックをもとに開発の方向転換を行う際，見積や変更管理プロセスを長時間かけて行うのではなく，優先順位に基づき素早く対応する能力をチームとして身につけることができる．また，UX コーチの参画を通して，ユーザー体験（UX）と呼ばれる利用者の視点を取り入れることで，システムの継続率と活用率が上がるとともに，素早くアイデアを検証することで，システムの方向性を素早く固めることができる．

「モダン開発基盤」は，現代的な開発技術を取り入れることで，高い生産性と品質を両立して変更要求を素早く取り込める基盤を活用すべきだができないという課題に対して，BIPROGY の長年の開発経験をもとに定めた参照アーキテクチャーである AlesInfiny Core を導入することで，検証済みのアーキテクチャーを活用できるサービスである．実装サンプルやトレーニングメニュー，ワンストップの相談窓口を設けることで，モダン開発基盤の活用を支援する．

「オブザーバビリティ」は，可観測性という用語で，複数の構成要素で成り立つシステムを統合的に監視することで即応性を高め，ビジネス指標と対応づけることで市場環境の変化を捉え素早く追従する次世代の監視の仕組みを提供するサービスである．オブザーバビリティにおいて重視すべき指標をヒアリングし，システム監視・ビジネス指標監視・ユーザビリティ監視など複数のツールを組み合わせて提供するとともに，運用・活用のノウハウを提供することができる．

「内製化」は，顧客の競争優位性のコアとなる業務をデジタルで再定義し，知識や技術，データを社内に確保することで素早い意思決定や方向転換ができる組織能力を獲得したいが，難しいという課題に対し，内製化の計画立案を支援し具体的なトレーニングメニューを提供するサービスである．具体的な開発案件に伴走型で支援しながら技術指導を行うことで，支援者のスキル定着を効果的に支援することや，従業員に対する教育・研修プログラムを提供し，全体のスキル底上げを狙うこともできる．あわせて，内製化に活用できるローコード基盤を提供している．

このように，AlesInfiny はインターネット普及以降の技術進化をパッケージとして提供し，組織の抱える様々な課題・ニーズに応じて段階的にモダン開発のノウハウを取り入れられるように構成されている．

6. 高度な AI を活用するための開発のあり方

人工知能（AI）の研究は 1950 年代からはじまり，その過程ではブームと冬の時代が交互に訪れている．探索や推論といった比較的単純な人工知能から，近年では「ビッグデータ」と呼ばれる大量のデータに基づき人工知能が知識を獲得する「機械学習」が実用化され，業務に活用されてきた．そして現在，「生成 AI（Generative AI）」技術を中心とした第四次人工知能ブームが生じている．これまでの AI と生成 AI の違いは以下の通りである．

- これまでの AI：汎用性が低く，事前に大量のデータで学習させなければならず，応用範囲が狭い
- 生成 AI：高い汎用性を持ち，業務に合わせた最低限の文脈（コンテキスト）を与えればよい．テキスト生成，要約，コード生成など応用範囲は幅広い

2022年11月に米国OpenAI社が発表したChatGPTは，チャット型インターフェースを通してあたかも人間と対話しているかのような体験を与え，爆発的な人気となった．ChatGPTのような言語を操る知性モデルは，大規模言語モデル（Large Language Models：LLM）と呼ばれる．現時点で世界最高性能を持つLLMはGPT-4（OpenAI社）である．その他，Google社が開発したLLMであるGeminiやMeta社が開発したオープンソースLLMであるLlama2など，競合も次々にLLMを用いた生成AIサービスを展開している．

LLMの性能はどれだけ人間に迫っているのかを端的に表すものが，GPT-4が米国の弁護士資格取得のための試験である「米国統一司法試験」において人間の合格者の上位10%に入るスコアを獲得した，というものである[5]．このような卓越した能力があれば，人間の仕事はすぐに奪われてしまうと心配する人もいる．しかし現在のところ，以下に示すLLMの技術的制約により人間の代替となることはない．

- 長期記憶を持たない：LLMは過去の質問内容を覚え続けることができず，一度に回答できる量に上限がある．
- 最新情報に追従できない：ある過去時点のデータに基づき学習を行うため，最新情報に追従するには外部検索する仕組みと組み合わせなければならない．
- 回答に自然な嘘が混じる：LLMは自身が持たない情報を「知らない」と答えずに，もっともらしい嘘をつくことがある．これをハルシネーション（幻覚）と呼ぶ．

LLMは人間の代替とはならないが，ある範囲においては人間と同等の回答を返すことができ，創造的な回答も行うことができる．業務の目的に応じてLLMから最適な回答を引き出すための質問（プロンプト）を考察し，業務に活用する新たなエンジニアリングの方法論が生まれた．これを「プロンプトエンジニアリング」と呼ぶ．BIPROGYも，様々なシステム開発の場面で活用できるプロンプト集を作成し社内にナレッジとして提供している．

高度なAIを活用する方向性としては，「提供するシステムに生成AIを組み込む」という方向性と「システムの企画・開発・運用に生成AIを活用する」という方向性がある．前者の方向性については，日本国内でも様々な技術検証が行われている段階である．

6.1　生成AIを組み込んだシステム

BIPROGYでも，Microsoft社が提供するセキュアなLLM実行基盤である「Azure OpenAI Service」を活用した金融機関向けカスタマーサクセス高度化プロジェクトを開始し，金融機関営業店窓口職員のコミュニケーションスキル向上の技術検証を実施している[6]．

ただし，提供するシステムに生成AIを組み込むことには様々なリスクがある．例えば，質問文として社内の機密情報を与えることで，生成AIの提供元に情報が漏洩するリスク，悪意のある質問文を入力することで想定外のサービス利用をされるリスク，著作権やパブリシティ権の侵害ケース等がある．これらのリスクを把握して適切に対処しなければならない．BIPROGYにおいても生成AIを安全に利用するためのガイドラインを策定した上で，業務に活用している．

6.2　生成AIを活用したモダン開発

モダン開発においては，積極的に生成AIを活用した開発を行うべきである．様々なシステ

ム開発の場面で活用できるプロンプト集を提供するだけでなく，開発プロセス全体を「AI ネイティブ開発」に切り替えることが肝要である．生成 AI を活用した開発の方向性は，表 1 のように分類できる．ただし，現時点で実用になっているものと検討段階のものがある．

表 1　開発運用への生成 AI の応用可能性

活用施策	2024年初期時点での活用可能性
コード生成	実用中 - GitHub Copilot等のコード生成サービスを活用し、開発に利用。コーディングの生産性向上に効果あり。
単体テストコード生成	実用中 - GitHub Copilot等の機能を用いて実装コードから一部テストコードの自動生成が可能。
コードレビュー	実用中 - GitHub Copilot Chat機能を用いてレビュー対象コードの分析と評価を実施しレビュー時間を削減。
設計書からの自動生成	実用に難あり - 設計書が既存のExcel等の場合、記述粒度が不均一であり利活用は難しい。一覧等均質な入力に整理する必要あり。
プロジェクト固有ナレッジ検索	実用に難あり - 既存ドキュメントをベクトルDBに登録しLLMから検索する仕組みを構築するも、データ品質を高める前処理がないと実用性が低い。

現在，システム開発における生成 AI の活用はコーディング補助や，叩き台としてのソースコードや仕様の作成など部分的なものに留まる．特に「既存システムに大量に存在するソースコードやドキュメントを読み解き，変更を取りこむ」という大きな現場ニーズを満たすことはできていない．その原因は，LLM の記憶容量には限りがあること，プロジェクト固有用語や知識を持たせることが難しいこと，様々な視点で記述されたドキュメントやソースコードを横断的に行き来しながら知識を獲得するための「眼や手」に相当する能力が乏しいことにある．これらの能力不足は将来的には克服されるだろうが，現時点では限られた LLM の能力をどう活かすかを考えるべきである．

現状は，具体的な情報が活用できるコーディングの工程での活用は進んでいるものの，より上流の工程で生成 AI を活用することや，工程間の成果物の変換などの活用は難しいといえる．これらの制約条件のもとで，生成 AI をモダン開発に活用するための施策を表 2 にまとめた．

表 2　生成 AI をモダン開発に活用するためのポイント

AI活用指針	説明
設計情報を単純なテキストにまとめる	設計書や各種仕様書などのドキュメントはExcel等バイナリデータとして扱われることが多い。これをMarkdown等のテキストデータとして作成することで、AIによる自動生成データとしてすぐに活用できる。
コードを小さく独立のファイルに分ける	LLMが一度に読み込めるコードの量には上限があるため、コードを目的別にファイルに分け、必要なファイルを参照することで精度を向上させることが可能。
あらゆる意思決定をドキュメントに残す	会議の内容をメモやホワイトボード・口頭で暗黙的に共有するのではなくドキュメントとして残すことで、将来的にナレッジベースとして活用できる。

生成 AI の得意領域は，具体化された成果物，特にソースコード領域である．そのため，上流工程の成果物もソースコードに近い形で表現することが望ましい．すなわち，SE とプログラマの領域が AI の登場により融合することが想定される．また，生成 AI の生成物に対しての説明責任はこれまで同様，作成者である人間に求められることになる．生成物に対する正しさを判断できる目利き力が強く問われる時代となるだろう．

LLM の発展と応用は日進月歩であり，手書きのメモやホワイトボードのスケッチを読み取れる画像認識や，音声でのインタラクティブなやり取りといったマルチモーダル AI と呼ばれる機能が提供されようとしている．マルチモーダル AI を用いることで，画面デザインの叩き台を作成し，デザインの叩き台からソースコードの雛形を作成することも一部では実用化している．

このような技術発展の成果を素早くチームの開発能力に取り込むことが肝要である．チームとして学習し目利き力を高めることが求められるが，アジャイルの特性である「継続的改善」の中で，これら新技術の調査・検証・自プロセスへの取り込みを行うことができる．人海戦術で大規模なシステムを構築する方法論から，AI を活用した少数先鋭のチームを活かす方向へ開発のあり方は変わろうとしている．

7. お わ り に

変化の激しい現代において，事業の継続と発展を目指すためには変化に追従できる組織に変革することが必須である．大規模な IT システムを資産として活用する時代から，必要最小限の IT システムを構築し，すばやく改修する時代になっている．エンドユーザである利用者の興味の変化と，オープンな時代の中での技術的変遷という 2 種類の変化に追従するためには，UX とアジャイルの技術を理解し，活用することがますます求められる．

時代の変化は，技術要素の積み重ねによっても生じている．「モダン開発」を理解し活用することで「作らない開発」を実践できるようになるが，素早く技術をキャッチアップするためには複数の案件で実証された開発基盤や開発手法を採用し，スキルの獲得を促すアドバイザーを傍に置くことが効果的である．BIPROGY はシステム開発の先駆者として長い歴史を持ち，顧客に伴走しながら価値追求を行ってきた．AlesInfiny はその経験の集大成であり，様々な組織課題を解決できるものになっている．

生成 AI 技術は今後ますます発展し，システム開発の自動化範囲を広げていくだろう．事業内容と課題を深く理解し，「作らない開発」を追求するチームが生成 AI など高度なモダン開発の技術を学習し，よりよい開発を目指していくことが求められる．アジャイルな考え方を実践できる組織作りを推進し，顧客が価値を提供し続けられるように貢献していきたい．

最後に，本稿の執筆にあたり，多くの方々にご助言とご指導を頂いた．この場を借りて深く御礼申し上げる．

＊ 1　AlesInfiny：BIPROGY が提供するアプリケーション開発運用の統合サービス．https://www.biprogy.com/pdf/news/nr_230324.pdf

＊ 2　SRE：Site Reliability Engineering（サイト信頼性エンジニアリング）の略語．Google 社が提唱する信頼性の高い本番環境システムを実行するための職務，マインドセット，エンジニアリング手法のこと．

＊ 3　VUCA：Volatility（変動性），Uncertainty（不確実性），Complexity（複雑性），Ambiguity

（曖昧性）の頭文字を取った略称で，「ブーカ」と読む．現代の組織やリーダーが直面する環境の不確実性と複雑さを表現した用語．

参考文献

［1］ World Competitiveness Yearbook, Swiss-based International Institute for Management Development (IMD), 1989-2023.
［2］ World Digital Competitiveness Ranking, Swiss-based International Institute for Management Development (IMD), 2023.
［3］ DX レポート 2.2（概要），経済産業省，2022 年 7 月
https://www.meti.go.jp/shingikai/mono_info_service/covid-19_dgc/pdf/002_05_00.pdf
［4］ DX 白書 2023，独立行政法人情報処理推進機構，2023 年 3 月
https://www.ipa.go.jp/publish/wp-dx/dx-2023.html
［5］ Daniel Martin Katz, Michael James Bommarito, Shang Gao, Pablo David Arredondo, GPT-4 Passes the Bar Exam, 2023.
［6］ 金融機関向けカスタマーサクセス高度化プロジェクト開始，BIPROGY 株式会社，2023 年 2 月 27 日
https://www.biprogy.com/pdf/news/nr_230227_2.pdf

※ 上記注釈および参考文献に含まれる URL のリンク先は，2024 年 1 月 19 日時点での存在を確認．

執筆者紹介

桝 田 秀 和（Hidekazu Masuda）
2003 年日本ユニシス（株）入社．公共系大規模開発案件の IT アーキテクトに従事した後，フロントエンド・モバイルアプリケーションの開発やスクラムマスターとして社内サービスビジネスのアジャイル開発を推進．現在は，BIPROGY が提供するアプリケーション開発運用の統合サービス「AlesInfiny」とアジャイルサービス「AlesInfiny Agile」を担当．