「作らない技術」の選び方

Choosing the Right Technologies for Aligning IT Strategies

瀬嵐雅樹，真野　悟

要　約　国内企業のIT投資戦略は，戦略的に投資する競争領域と，企業活動を維持する機能を提供する非競争領域に大別される．さらに，競争領域は，顧客要求の変化に迅速に対応する攻めの領域と，企業活動のために差別化を図る守りの領域に分けられる．この三つの領域に，技術の観点として「効率的に作る技術」と「再利用する技術」を併せて整理することが，企業の目的とIT人材戦略に応じて技術を選択する時の指針として有効である．やみくもに最新技術を追うのではなく，技術を使える人材やコスト面も踏まえ，将来的にも主流であると考えられる技術を選択する「技術の目利き力」が重要である．

Abstract　Domestic companies' IT investment strategies can be broadly divided into competitive areas in which they invest strategically and non-competitive areas in which they provide functions to maintain corporate activities. Furthermore, competitive areas can be divided into offensive areas, which quickly respond to changes in customer requirements, and defensive areas, which aim to differentiate corporate activities. Organizing these three areas together with "techniques for efficient production" and "technologies for reuse" from a technology perspective is effective as a guideline when selecting technology according to a company's objectives and IT human resources strategy. Rather than blindly pursuing the latest technology, it is important to have "the ability to discern technology" to select technology that is considered to be mainstream in the future, taking into consideration the human resources who can use the technology and the cost.

1. はじめに

デジタルトランスフォーメーション（以下，DX）が企業成長のカギとされる現代では，企業活動においてビジネス価値や社会課題などのビジネス戦略が重要なテーマとなっており，ITはその実現手段として同じく重要事項となっている．企業のIT戦略の観点では，新規デジタルビジネスの創出，既存システムのモダナイゼーション，内製化，など取り組むべきことが多岐にわたっている．

システムを構築する技術に関しては，今までのプログラミング言語による開発効率化・品質向上への取り組みだけでなく，システムを構築するための技術選択肢は多岐にわたってきているが，どのような要求にも対応できる万能な選択肢は存在しない．そのため，システムの課題と構築する目的を明らかにし，適材適所で新しい技術と手法を導入し，効率的な開発・運用をしていくことが重要になってくる．また企業におけるIT技術者の育成・調達方針とも密接に関係するため，人材育成戦略の一部としても適切な技術を選択しなければならない．

経済産業省は，持続的な企業価値の向上を図る施策として，デジタルガバナンスコード[1]を提唱し，ITとビジネスを一体的に捉えることを推奨している．企業はITシステム・デジタル技術活用環境の整備として，具体的に利用する技術・標準・アーキテクチャ，運用，投資計画

等を明確にすることが望ましいとされている．そのため，2023 年末時点のシステム構築に関する技術を俯瞰的に捉え，利用シナリオと適した技術の組み合わせを紹介することは，IT 戦略に応じた効率的な開発と運用を進めていくために有用な情報となると考える．

本稿では，注目されている新しい開発技術を含めた整理と技術の選択方針について考察する．まず 2 章で国内の IT 開発を取り巻く現状と課題を述べ，3 章で技術の検討指針と具体的に利用する各技術との関係を整理し，4 章で各技術の概要を解説し，5 章で「効率的に作る技術」「再利用する技術」における技術の選び方を利用シナリオと合わせて解説した後，6 章で将来に向けた展望を述べる．なお，技術自体の詳細や歴史的な変遷の詳細についての解説は本稿の範囲に含めないこととする．

2. 国内の現状と課題

本章では，国内の DX 実現に向けた IT への取り組みの現状と課題とともに，システムを構築する技術を検討する際の観点について俯瞰的に述べる．

2.1 国内企業の IT への取り組み状況

企業を取り巻くビジネス状況は不確実性が高く予測が困難と言われている．未来に向けたビジネスの方向性を踏まえ，システムが採用する開発技術を企業のシステム特性に適合するように定めるためには，技術の動向を正しく理解し選択することが一層重要となってきている．

独立行政法人情報処理推進機構の「DX 白書 2023」[2]によると，システムの開発手法・技術の導入目的は「ビジネスニーズの変化に対する柔軟な対応」「ソフトウエアの品質向上」「ソフトウエアの生産性の向上」が上位にあるとともに，国内と米国で傾向に大きく差異がある「開発コストの削減」「インフラ運用コストの削減」「レガシーインフラの刷新」といったコスト削減を目的とすることも重要視されている．これらは，ビジネスニーズの変化に柔軟に対応できるシステムの構築を進めるとともに，企業の戦略に基づいて老朽化した既存システムの適切な移行方針と技術を検討することが重要だということを示している．

2.2 企業戦略と IT 投資戦略

国内企業の IT 投資戦略はビジネス要求や目的に応じて，戦略的に投資する競争領域と，企業活動を維持する機能を提供する非競争領域に大別される．さらに，競争領域のシステムは，攻めの領域として顧客要求の変化に迅速に対応し差別化を図る SoE[*1]/SoI[*2] などのシステムと，守りの領域として SoR[*3] に属するが企業活動のために差別化を図るシステムに分類される．企業の IT 投資は競争領域に対して積極的な投資を検討すべきであり，変化の少ないシステムは非競争領域として開発運用コストの削減と効率化を目指すことが重要である（表 1）．

国内企業の IT 投資を IT 予算配分の観点から見た場合，JUAS による調査[3]では，予算内訳は既存ビジネスの維持・運営が約 8 割を占めている状況が継続し，売上高に占める IT 予算比率も業種全体で，2022 年度は 1.24%（トリム平均値），前年度比 0.04 ポイント増と過去数年に渡り大きな変化がないことからも，限られた予算の中でより効果的な予算配分を行うことが重要となる．そのためにも，これからのシステム企画においてはビジネス戦略と IT 投資戦略を基に，システムを競争領域（攻めの領域と守りの領域），非競争領域に分類し，それぞれのシステムに対する開発保守方針をロードマップ化することが推奨される．

表 1　競争領域・非競争領域と企業の IT 投資戦略

領　域	目的・対象システム	IT 投資戦略
競争領域（攻めの領域）	顧客要求の変化への迅速な対応 ・新規領域（SoE，SoI）	・新規システム開発への投資 ・新技術/方法論の採用
競争領域（守りの領域）	企業活動のために差別化を図る ・一部の既存領域（SoR）	
非競争領域	変化が少なく，企業活動を維持 ・既存領域（SoR）	・コスト削減策の検討 ・できるだけ作らずに実現

2.3　技術選択における課題

競争領域と非競争領域をシステム開発の観点で考えると，仕分けされた領域の目的を実現するために，最適な技術を選択しなければならない．特に競争領域のシステムは差別化を図るためにも，最新の技術や手法の積極的な検討が求められる．非競争領域においてもコスト削減の手段として，クラウド化の推進をはじめとしたモダナイゼーションなど，新しい技術に取り組むことも重要である．

一方，既存システムの開発現場においては，開発生産性・品質を維持するために手慣れた開発手法や技術を継続利用することに重点が置かれることにより，新たな技術や手法への取り組みの優先度が下がり，モダナイゼーションに向けた十分な検討が進んでいないことも推測される．このような状況の中で最新の技術を検討する際に注意すべき点は，技術が過渡期な状況においては選択肢が多く，かつ技術の変遷も速いということである．

システムを将来的にも継続して利用するためには，技術の動向とデファクトスタンダードを正しく把握し，目的に合致した将来性のある技術を選択する視点が重要である．この視点が欠如すると，ビジネスニーズと技術が一致しないまま利用する，技術を過大評価しメリットやデメリットを考慮せず目的と手段が逆転する，人材育成計画と要員スキルがマッチせず技術を使いこなせない，といった状況に陥り，目的を実現できず無駄な投資となる弊害が予測される．

経済産業省の「DX レポート 2.1」[4]では，DX に用いる技術を獲得しようとする企業に対して，パートナーの専門家が技術や組み合わせを提案することを推奨していることからも，技術選択を企業単独で行なうことは難易度が高いと考えられる．

2.4　IT 技術者に求められる姿と課題

IT 技術者確保の観点では，主に DX 推進を目的とした IT 技術者の確保は，戦略に基づいたシステム開発の実行において重要な事項であるが，求められる技術者の充足度において「量」「質」ともに大幅に不足する傾向が拡大している．「DX 白書 2023」[2]によると，人材の「量」の確保について「大幅に不足している」が 2021 年度の 30.6％から 2022 年度は 49.6％に増加，「質」の確保についても同様に 30.5％から 51.7％と増加している．

こうした IT 技術者不足は，技術の適用範囲の拡大，選択肢の増加，難易度の高度化などに対応するための自社要員の育成と，外部パートナー調達の両面で慢性的に発生している．戦略的にシステム開発に取り組むためには，生産性向上と品質を両立させる手段を獲得しなければならない．効率よく作る技術と提供される技術を再利用して都度開発しないといった観点での取り組みが求められる．

また自社要員による内製化への指針として，システムの特性に着目し「開発規模が小さい」「業務が複雑でなく難易度が低い」など対象とするシステムの指針を定めた上で，内製化に適した技術を選択し，自社要員の育成にも合わせて取り組むことが肝要である．

2.5　開発技術の分類

ここまでは国内企業の IT への取り組み状況と課題について述べてきたが，こうした課題を解決していくための開発技術をシステム構築の手法の観点で見た場合，以下のように，様々な手法と関連する技術が存在する．

- 従来の開発言語を利用したいわゆるスクラッチ開発
- ローコード・ノーコードなどツールを利用した開発
- SaaS・パッケージなどのサービスの活用

また今後の開発の在り方を変える可能性を持つ新しい技術としてコード生成 AI を活用する AI ネイティブ開発の手法が注目されている．

企業にとって，戦略実現に向けたシステムの仕分けと開発方針の検討をトップダウンで進めることが重要であるとともに，業務部門などの社内ユーザー自身が業務効率化を目的として取り組む EUC などのボトムアップのアプローチも合わせて考慮することが望ましい．これらの技術はそれぞれの特徴と目的によって，前節で述べた，効率よく作る，提供される技術を再利用して都度開発しない，という観点で分類することができる．次章では「効率的に作る技術」と「再利用する技術」として具体的に解説する．

なお，本稿では実現方法としての技術について述べるため，アジャイルなプロセスには触れないが，競争領域において，アジャイルな取り組みによりビジネスニーズの変化に柔軟に対応し，仮説検証型で顧客や市場の要求を見定め，状況に応じて求められるものを迅速かつ柔軟に取り込み，不要なものは開発しないアプローチも重要な要素である．

また，システム全体の開発・実行・運用にはインフラ技術が必須であるが，本稿では業務機能を構築する技術にフォーカスするため多くは触れない．インフラ技術は，従来のオンプレミス環境しか選択肢がない状況から，現在ではクラウド環境や DevOps 技術が普通に利用されるようになっており，以降の章でどの技術を選択する場合においても，適切なインフラ構築を前提としている．

3.　作らない技術の検討指針

本章では，技術を選択する際の考え方と選択対象とする技術を俯瞰的に整理する．目的に応じて複数の選択肢から技術を使い分けるには，IT 投資戦略に挙げた「競争領域（攻めの領域・守りの領域）」「非競争領域」の該当するシステムに基づき，新規ビジネス立上げ時の必要最小限の機能を備えたシステムなのか，基幹業務系のシステムなのか，などシステムの特性やライフサイクルも考慮して検討することが望ましい．

新しい技術の登場や既存技術の成熟化を経て採用技術の選択肢が増えており，選択基準も内製化方針や IT 人材育成方針を踏まえて複雑化しているため，単純に技術観点で評価するだけでは正しい選択をすることが難しくなっている．

IT 投資戦略に挙げた三つの領域と，技術の観点として「効率的に作る技術」と「再利用する技術」を併せて整理することが，企業の目的と IT 人材戦略に応じて技術を選択する時の指針として有効である．以降の節で，その考え方と対応する技術について論ずる．

3.1　効率的に作る技術と再利用する技術

作る技術，作らない技術という区分けで語る場合，「誰が」「何を」作る対象としているかに注意すべきである．作る技術とは，IT 技術者が従来通りスクラッチ開発をするケース，つまりプログラミング言語を使い一からコーディングするケースを指すことが多い．

本稿では，作らない技術として，コーディングに限らずシステムを実現する際の技術全般を対象とし，「誰が」には開発者だけでなく非 IT 技術者も含め，「何を」にはコーディングを代替する技術，再利用する技術などを含める．

具体的な技術として，企業の IT 戦略，IT 人材戦略で積極的に採用されている技術である OSS フレームワーク，ローコード・ノーコード，RPA[*4]，SaaS・パッケージの四つを，これからの取り組みとなる AI ネイティブ開発，マイクロサービス，オープンソース・インナーソースの三つを選択した（技術の概要や動向は 4 章で説明）．これらは，競争領域のスクラッチ開発で選ばれる生産性向上技術，非競争領域の定型業務で選ばれる再利用技術，内製化のリスキリングで選ばれるスキル習得コストが低い技術，あるいは企業全体での再利用促進のために選ばれるアーキテクチャや組織的な取り組みなどである．

図 1 は本稿で取り扱う技術をまとめたものである．企業 IT 戦略の観点にて競争領域ではスクラッチ開発・OSS フレームワークの活用とシステム特性に応じたローコードの活用，非競争領域では SaaS・パッケージを位置付けることができる．以降の節では，効率的に作る技術と再利用する技術について解説する．

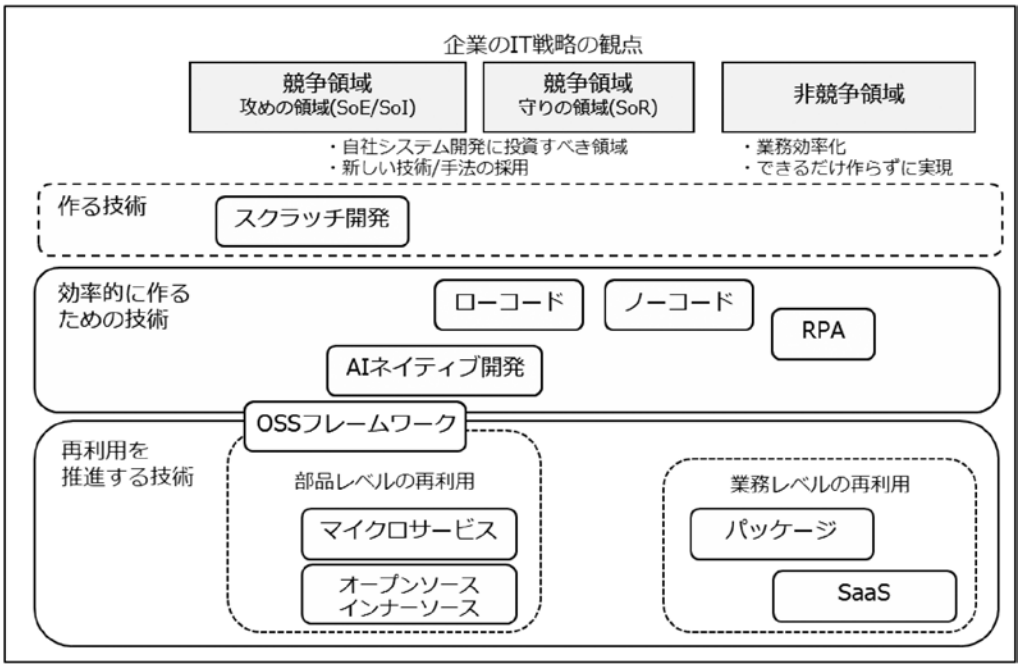


図 1　本稿の対象とする技術

3. 2　効率的に作る技術

2010 年代前半までは，企業が利用する規模のシステムにおいては，現実的には作る手段はスクラッチ開発が主な選択肢であった．また，部署レベルでの EUC の分野では，表計算ソフトでのマクロの利用までであった．前節で選択肢に挙げた技術は当時から知られており，一部の開発では試行されていたが，発展の途中でありデファクトスタンダードとなる技術ではなかった．信頼性の面でも不十分であり，一般的な選択肢ではなかった．

2010 年代後半以降，各技術で注目すべき製品の登場や進歩により認知度が向上し，選択肢が広がっている．BIPROGY 株式会社（以下，当社）においても検証を踏まえて実用レベルと判断した技術については，実案件での採用が増加している．またプログラムを作らない領域の技術を，その特性を活かせるシステムに適用することにより，IT 人材の質・量が不足している課題の一部解消につなげる効果も期待できる．

効率的に作る技術の採用によるメリットの例を挙げる．

- プログラムを作る部分を削減できる，あるいは無くせること
- プログラミングスキルを持たない要員がアプリケーションを実装できること
- 素早く開発でき，開発期間を短縮できること

スクラッチ開発を高度なスキルと相当な作業量を要する技術と捉えたときに，効率的に作る技術として，従来型のプログラム開発作業を効率化しながら品質を維持するために「OSS フレームワーク」を利用することになる．また，コードを書かない技術として，ビジュアルなインターフェースでアプリケーションを構築できるツール「ローコード・ノーコード」，人的作業を自動化する「RPA」に加え，注目技術であるコーディング作業そのものを自動化する「コード生成 AI」が選択肢に挙げられる．なお，「コード生成 AI」については，開発のあらゆる側面で AI 適用を検討する発展的な取り組みと捉え，次章では広く「AI ネイティブ開発」として取り上げていく．

3. 3　再利用する技術

再利用する技術を考えた場合，再利用の粒度によって視点が変わってくる．最小の粒度として共通部品やフレームワークを再利用する視点は前節の効率的に作る技術を下支えし，ビジネスレベルの粒度で再利用する視点では業務自体または一部の汎用機能を SaaS・パッケージを採用して実現するなど対応領域自体が異なってくる．

業務自体の再利用は非競争領域にて SaaS・パッケージを採用し，システムに業務を合わせる形で活用されるが，対象となる基幹業務のビジネスプロセスをそのまま当てはめようとしてカスタマイズが発生する場合も多く見受けられる．業務の再利用とカスタマイズは相反することが多く，カスタマイズ部分が肥大化して保守が困難となる，バージョンアップの際に互換性が失われるなどの課題があることに注意してほしい．

再利用する技術の採用によるメリットの例を挙げる．

- 同じ処理を何度も実装する手間が省けること
- 過去に利用され，問題なく動作しており，信頼性が高い実装を利用できること
- 作るときから再利用する設計にすることで，効率よく再利用できること
- 共同開発体制により，広い範囲での再利用や，生産性向上が期待できること

再利用する技術としては，他社が提供するアプリケーションやサービスを利用する「SaaS・パッケージ」があり，業務そのものへの適用や部品として組み合わせてシステムを実現することが考えられる．企業内での技術標準化を目的とした場合「OSS フレームワーク」は再利用する側の技術としても位置付けられる．

また将来的なシステム拡張や分割時に備え，再利用することを目的としてアーキテクチャを設計する「マイクロサービス」も注目されている．さらに「オープンソース・インナーソース」は技術そのものではないが，ソースレベルでの再利用と流通を推進するための組織的な取り組みとして，今後さらに重要性が増すことが想定されるため，技術と合わせて紹介する．

4. 効率的に作る技術，再利用する技術の動向

本章では効率的に作る技術，再利用する技術を検討する際に選択肢となる，それぞれの技術について概要と動向を説明する．

4.1 OSS フレームワーク

競争領域における効率的に作る技術として，OSS フレームワークがまず挙げられる．近年のシステム開発では，生産性や品質の向上を図るため，アプリケーションフレームワークの適用が一般的である．商用あるいはシステム開発ベンダー独自のフレームワークは，特定ベンダーへの依存度の高さ，インターネット等で公開されている利用技術情報量の少なさやこれに起因する習得コストといった課題が懸念される．一方で，OSS フレームワークは，生産性や性能などの品質が強化されていることから適用が増えている．また近年の動向として，技術をオープンにする傾向が進んでおり，従来は商用だったフレームワークが OSS として公開されるようになったケースも増えている．

どのような OSS フレームワークを採用するかについては，システム要件に合致することは前提として，開発時に利用できる技術情報が多いかどうか，潜在的な問題やバグが少ないか，脆弱性などの問題が発生した際に適切に対応されるかといった観点から判断するのが賢明である．これらの観点から，同様の機能を提供する OSS フレームワークの中でも，デファクトスタンダードであるもの，開発コミュニティが活発であり積極的に更新されているもの，サポート期間が長いものを採用することが適切である．ただし，フロントエンド技術などデファクトスタンダードの切り替わりが速い領域もあるため，OSS の動向を注視し，最新の情報にアップデートしていくことも重要である．

4.2 ローコード・ノーコード

競争・非競争領域に跨り，効率よく作る技術としてローコード・ノーコードがある．ローコード・ノーコードの普及度は高まっており，活用目的は大きく次の二つに大別できる．業務部門の利用者による簡易なデジタイゼーションで業務改善を狙うボトムアップ型と，IT スキルを備えた開発者がシステム群の開発の高速化を狙うトップダウン型である．後者は，システム間連携やデータ利活用，機能の再利用性等を意識した一定の複雑性を伴うシステムの開発で採用されることが多い．それを実現するためのトランザクション設計，例外パターン設計，データの正規化，品質確保戦略策定などの開発スキルもあわせて要求される．ボトムアップ型とトップダウン型の両方に対応する製品は存在しないため，目的別に製品を選択することが望ましい．

ボトムアップ型の活用領域は自部門の業務改善を目指すため，特定の業務に閉じたスコープとなる．トップダウン型の主な活用領域は，競争領域と非競争領域それぞれで周辺システム群がターゲットとなりやすい．差別化領域では，先端技術で実現した仕組みを顧客や従業員が利用するシステム，新業務と既存業務を連携するシステムがターゲットとなる．非差別化領域では，SaaS や ERP など作らない手段と業務のギャップを外部で吸収する手段として，ワークフローシステムなどが考えられる．スクラッチ開発と比較すると，この様な周辺システム群はボトムアップ型ではなくトップダウン型の活用目的を主軸とした製品が多い傾向がある．

4. 3　RPA

RPA は非競争領域にて，業務プロセスやタスクの処理手順を自動化し，業務の効率化を実現するためのツールである．業務に携わる担当者が自ら手元の業務を効率化でき，既存の業務アプリケーション改修が基本的には不要となるため，導入コストの規模が比較的小さく，適用のスピード面で優れている．一方でシステム開発やパッケージ導入等と比較するとスケールメリットは生じにくい．大規模な DX を進める前段階で十分な余剰工数を生み出すため，従業員の DX 意識を醸成するための利用が効果的である．

十分に浸透した技術であるが，RPA の先行ベンダーは，UI 認識の精度向上や対象範囲拡大，AI-OCR，iPaaS 等の関連機能の追加や他製品との連携機能を強化し機能充足を図る傾向にある．一方で後発ベンダーは，機能を絞った上で価格優位や手軽さを押し出している．今後は AI との統合利用によって，さらに高度な自動化の実現が期待されている．

4. 4　SaaS・パッケージ

SaaS・パッケージは，非競争領域にてビジネスレベルの粒度で再利用する手段として，「早く簡単にビジネスを実現したい」という顧客課題を実現する選択肢となっており，広い業務分野において BtoB から BtoC まで多岐にわたるサービスが提供されている．

SaaS は大きく以下の二つに分類される．

1）業務 SaaS：経費精算や勤怠管理といった業務そのものを行う
2）部品 SaaS：SMS 送信や ID 認証，オンライン決済など業務の中の一機能の役割を果たす

業務 SaaS は，特定の業務を実現するという点ではパッケージと似ている．業務 SaaS とパッケージのいずれを利用する場合でも，ビジネスプロセスに合うようカスタマイズするのではなく，ビジネスプロセスの方を業務 SaaS やパッケージに合わせるべきである．しかし，独自かつ複雑なビジネスプロセスをシステム化することが多い基幹系業務領域では，ビジネスプロセスを変更することは難しい．他システムとのデータ連携，ユーザーごとの細かなアクセス権限制御やデータ種別による処理フローなどを考慮しなければならない．

一方，部品 SaaS はあくまで業務の一部の機能を担うものであるため業務を問わず，認証や監視などの部分で利用することができる．部品 SaaS には様々なサービスが存在する．複数の部品 SaaS を組み合わせて汎用的な機能を実現し，ビジネスのコア部分の開発に注力することで，実現したい新たなビジネスを低コストで早く構築できるようになる．

4.5　AI ネイティブ開発

競争領域にて効率的に作るための最新の取り組みである AI ネイティブ開発は，人工知能（AI）との協業を前提として，ソフトウエア開発の生産性と品質を向上させるアプローチである．この傾向は今後加速すると考えられるため，企業の競争力が問われている．

ChatGPT や GitHub Copilot などの生成 AI サービスは，ソフトウエア開発における様々なタスクを支援することができる．ChatGPT は技術的な内容を含む幅広い指示や質問に回答できるため，設計からソースコード生成に至るまで，幅広く活用できる．GitHub Copilot はコーディング支援に強みがあり，指示を書いたら Tab キーを押すだけでコードが完成することもある．ただし，ChatGPT や GitHub Copilot の AI は特定の過去時点における公開情報を基に学習しているため，最新の情報や公開されていない文脈（例：社内用語）に則した回答には限界がある．また，通常は入力のサイズに制限があるため，大量の入力を伴う複雑な文脈を前提とした質問をすることはできない．今後，これらの限界を克服する仕組み・サービスの登場や，画像や音声などテキスト以外のデータを入力することができるようなマルチモーダル化，そしてモデル自体の能力向上により，開発者の生産性や利便性が更に向上すると予想される．

さらに適用する場面を拡大し，新規開発時だけではなく，既存システムのモダナイゼーションにおいて，検討時における設計ドキュメントの生成，移行時に旧バージョンの環境で作成されたプログラムの最新化などにも活用の余地があると考える．

このように，生成 AI の進化は，ソフトウエア開発における開発者体験を根本的に変えつつある．AI との協業を前提とした新しい開発プロセスと文化を取り入れた，AI ネイティブ開発の推進が求められる．

4.6　マイクロサービス

競争領域においてビジネス成長に備えた再利用アーキテクチャとしてマイクロサービスを取り上げる．アジャイル開発やクラウド技術の発展に伴い，臨機応変に対応できるアーキテクチャが求められた．そこで登場したのがマイクロサービスアーキテクチャである．すべての機能が単一のアプリケーションで完結する従来型のモノリシックアーキテクチャに対して，マイクロサービスアーキテクチャはサービス単位でアプリケーションを分割し独立させ，サービス間で通信することにより業務機能を実現する（図 2）．

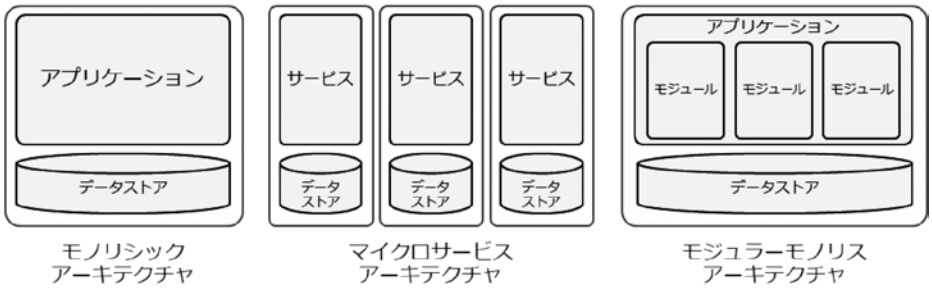


図 2　アーキテクチャ比較図

サービスが独立することによる主なメリットを以下に挙げる．

1）変更容易性

各サービスが独立しているため変更に対する影響範囲が小さくなり，変更を加えやすい

2）業務再利用性

新機能追加時などに既存サービスの再利用が容易になる

3）スケーラビリティ・可用性

サービスが独立しているため特定のサービスのみスケールでき，耐障害性が向上する

反面，マイクロサービスアーキテクチャはビジネスの複雑さと求められる技術的要件などの課題も存在する．例えば，開発初期段階でビジネスコンテキストを正しく分割することは非常に困難であり，サービス要件の変更に伴い継続的に再構築することが求められる．またサービスの分割に伴いデータストアを分割すると，各サービス間でのデータ整合性を考慮しなければならず実装難易度が高くなる．こうした要因によりマイクロサービスアーキテクチャを導入することは必ずしもビジネスを成功に導くとは限らない．

マイクロサービスアーキテクチャが持つ課題に対して，注目されているのがモジュラーモノリスアーキテクチャである．モジュラーモノリスアーキテクチャでは，単一アプリケーション内で各サービスをモジュール単位に分割するためモジュール単位での開発ができるようになり，単一データベース上でデータを管理するため整合性を担保して実装することが容易となる．モノリシックアーキテクチャ同様に，デプロイラインやスケーリング単位は統一される側面もあるが，適切な単位で分割されているモジュールは，一つのアプリケーションとして独立させることができる．

このようにアーキテクチャレベルでの再利用を検討する際は，プロジェクトの要件や目標を総合的に検討し，適切なアーキテクチャを選択することが重要である．

4.7　オープンソース・インナーソース

部品・フレームワークの粒度でシステムを効率よく開発する手段としてオープンソースの利用は欠かせないものとなっている．本節ではさらにソフトウエアを再利用することを目的とした組織的な取り組みについて説明する．オープンソースはシステムで共通利用できる高品質なソフトウエアを開発・配信する手法として発展している．利用者はソースコードの閲覧や，ニーズにあわせた改良ができ，システムで使う機能を取り込むことで生産性の向上が期待できる．オープンソースの提供者はソフトウエアの改良を受け取る機会が創出でき，ユーザーの獲得や市場の拡大につなげられる．企業のオープンソースへの投資は，「社会に貢献する企業」としての宣伝効果も期待できる．一方，ビジネス価値のある技術をオープンソースにすると，ビジネスの模倣が成立してしまうため，企業の利益と相反する恐れがある．

それに対してインナーソースは，企業内でのみソースコードを共有し，生産性向上を目指す取り組みである[5]．オープンソース同様の開発方式を採用するが，公開範囲が限定されるため，企業機密の取り扱いもできる．また開発に参画する際の心理的ハードルも低くなる．ただし，ユーザーの獲得など，一部のオープンソースのメリットは享受できない．

オープンソースの利用が浸透したのは，導入難易度の低さと，品質の高さを両立したからである．機能性が高いことに加えて，誰でも利用できるよう配慮されたドキュメントの整備や，開発状況の見える化が図られている．これらの実現には，開発組織や開発プロセスの整備が不可欠である．

オープンソースやインナーソースの活用には，著作権とライセンスへの理解が必須である．一般的に開発者が著作権を持ち，利用者はライセンスに従った利用許諾を与えられる関係となる[6]．オープンソース提供時は，OSI（オープンソースイニシアティブ）で承認されたライセンス[7]から，提供者の目的に合致するライセンスを適用することが望ましい．適切なライセンスを適用しないと，利用が広がらず，公開する意義が失われることもある．インナーソース提供時の制約は少なくなるが，複数法人を抱える企業体でのコード共有には，法的な契約を要する．また顧客にシステムを納入するビジネスを展開する場合，著作権を開発元に留保したまま利用権を与える契約を結ばねばならない．

オープンソースやインナーソースは，システムの品質と生産性の向上に役立つものである一方，適切なライセンス選択や著作権に関する対応など，検討事項は多岐にわたる．ビジネス価値を維持しながら，オープンソースやインナーソースの活用を推進しなければならない．

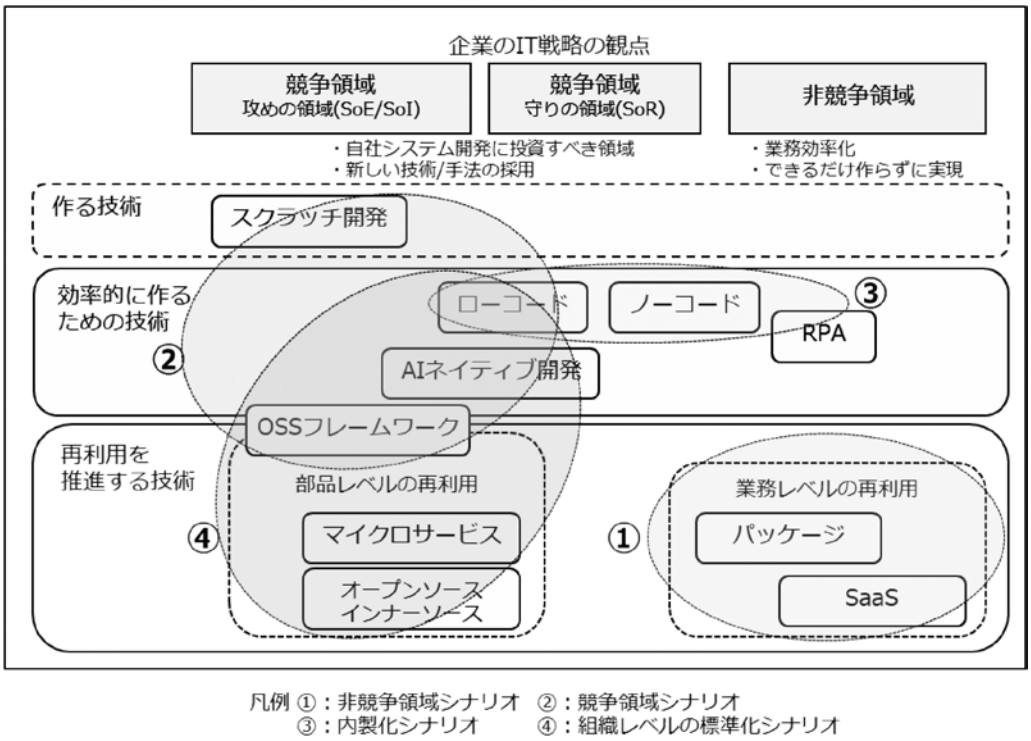
5. 作らない技術の選び方

前章までで挙げた，IT 戦略に基づいたシステムの分類と，効率よく作る技術，再利用する技術の観点を組み合わせることにより，「利用シナリオ」すなわち企業の IT 戦略・人材育成戦略と，選択する技術の対応付けが見えてくる．本章では利用シナリオを基に具体的な技術の利用イメージをまとめることにより，作らない技術の選び方を明らかにしていく．

5.1 シナリオと技術の選び方

技術の選び方を考える際には，まず企業のビジネス戦略と IT 戦略に基づき，対象となるシステムの位置付けを明確にする．そのため利用シナリオは IT 戦略・人材育成戦略に沿ったシステム化の方針を中心に設定し，シナリオに対して，ここまで個別に論じた「効率的に作る技術」と「再利用する技術」がどのように活用されるかを考察する．

具体的なシナリオとして「非競争領域」「競争領域」「内製化」「企業レベルの標準化」を設定することとし，「非競争領域」「競争領域」は投資戦略に基づいた最適な技術の選択肢について，「内製化」は人材育成戦略との関係を踏まえた考え方について，「企業レベルの標準化」は企業や外部パートナーを含めた再利用への組織的な取り組みについて論ずる（図 3）．



凡例 ①：非競争領域シナリオ　②：競争領域シナリオ
③：内製化シナリオ　④：組織レベルの標準化シナリオ

図3　シナリオ概観図

5.2　非競争領域

非競争領域のシステムは，企業活動を維持するために欠かせない業務だが，ビジネスの差別化には直結せず，変化も少なく安定している業務領域のシステムということができる．具体的には人事・給与・会計などの業務を挙げることができ，運用の効率化・コスト削減が求められる対象となる．2章で触れたIT投資の観点からも，限られたIT予算の内訳を既存ビジネスの維持・運営から，できるだけ競争領域にシフトするために，非競争領域へのコスト配分を削減する取り組みが求められる．そのため，システムを従来通りに作るという発想から，できるだけ作らずに実現する考え方に切り替えるべきである．

システムを作らずに実現する手段としてはSaaS・パッケージの採用が一般的である（5.1節 図3①）．業務そのものにSaaSを適用し，システムに業務を合わせることにより導入時のコスト・期間を削減できるメリットを生かすことが重要である．パッケージ製品では仕様のカスタマイズもある程度許容されることもあるが，SaaSは他社含めてマルチテナント型で提供されることが一般的であるため，業務機能に関する仕様変更の自由度は低くなる．一方で，利用企業の要望が製品仕様に反映されているため，対象業務の機能はほぼ提供されていると言えるだろう．

非競争領域で対象となるシステムには，企業独自のビジネスプロセスが組み込まれている場合もあるが，現行仕様を踏襲する目的でのカスタマイズは注意すべきである．できるだけ作らずに実現する手段に対して，カスタマイズは改造部分に保守が生じるためコスト削減の効果が薄れる，パッケージやサービスのバージョンアップ時に非互換が発生して代替策の追加開発が発生するなどリスクも含んでいる．

できるだけ作らずに実現する観点では，ビジネスプロセス上，SaaS やパッケージでは不足する機能をカスタマイズするのではなく，プロセスやシステム連携などを調整して，不足する部分をシステムの周辺に補完機能として別途開発することを検討することが望ましい．このような周辺開発の実現方法としては，SaaS やパッケージの導入と合わせスピード感をもって提供できる，ローコードによる開発が選択肢となる．

5.3　競争領域

次に，競争領域でのシステム開発のシナリオについて考察する．コアコンピタンス実現のために推進すべき領域で，ビジネス的な重要度が高く積極的な投資の対象となる．特に攻めの領域では，イノベーションに向けたビジネスアイデアが重要であり，他社と差別化された新しい機能を実現するため，ビジネスに合わせてシステムを作る方針となる．このため，実現するシステムに制限が少ない技術を選択することが重要となり，自由度の高い設計や機能を実現できるスクラッチ開発を採用することが多くなる．また，市場に投入するスピードを速め，市場からのフィードバックを素早く改善につなげることがビジネスの成功に直結するため，スクラッチ開発の生産性を向上できる技術がマッチする（5.1 節 図 3 ②）．再利用する技術についても，部品の仕様が全体の機能実現を阻害しないように，業務に影響を与えない程度の小さな部品単位での再利用が求められる．

この領域では，顧客接点のシステムが多く，業務機能だけでなくフロントエンドやモバイルアプリでの UX の差別化も重要になってくる．スクラッチ開発の生産性向上のため OSS フレームワークを採用するケースがほとんどだが，フロントエンドやモバイルアプリの技術トレンドは特に移り変わりが激しいため，表現力が高く UX の向上を図れる最新のフレームワークや開発環境を積極的に採用することが有効である．

一方，守りの領域では，SoR に代表される，基幹業務の大規模システムで長期間稼働するケースが多く，より品質と信頼性が重要となる．守りの領域でも部品単位の再利用を行うが，最新の OSS フレームワークよりは，長期的なサポートが望めるデファクトスタンダードを選択することが有効である．守りの領域のシステムはインターフェースを API 化して疎結合な作りとしておくことで，他のシステムへのデータ提供をスムーズに行うことができる．長期安定稼働が多い守りの領域と，市場への継続的リリースが特徴となる攻めの領域でリリースサイクルに違いがあるが，疎結合にしておくことにより攻めの領域のシステムリリースへの影響を小さくできる．

今後取り組むべき技術として，AI ネイティブ開発の活用を検討する．具体的には，攻めの領域では，画面デザインからのプロトタイプシステム自動生成，ソースコードからのテストコード自動生成などにより，生産性・品質のさらなる向上を狙える．守りの領域では，言語やフレームワークのバージョンアップ対応の事例などが報告されており，従来は時間とコストをかけていた作業がより少人数でできるようになり，要員を生産的な仕事にシフトできる可能性がある．なお，システムの中心部分をスクラッチ開発する場合も，システム間の連携などの周辺システム群については，開発の効率化のためローコードを採用することも有効である．

5.4　内製化

続いて，内製化開発を行うシナリオについて考察する．内製化に向けてはまず企業の育成計

画や IT 技術者調達の方針が重要となる．攻めの競争領域で新規サービスなどの開発を行う際に内製化を検討する場面を考えてみる．5.3 節で述べたようにスクラッチ開発となる場合が多く，スクラッチ開発で使用する技術を内製で獲得することになるため，スキル習得コストが高くなる．このため，自社で育成するのか，外部から IT 技術者を採用するのか，外部の IT ベンダーを活用するのか，自社要員スキルの現状を踏まえた上で将来的な内製化の目指す姿について方針を決めることが重要である（表 2）．

表 2　内製化における人材育成・調達戦略

人材育成・調達戦略	育成・調達アプローチ
自社要員育成（低スキル）	リスキリングによるローコードツールなどの活用検討
自社要員育成（高スキル）	要求されるスキルを持つ技術者を社外から採用する．社外認知度向上のためのプロモーションなどと合わせて実施
	外部パートナーが開発に参画・伴走することで，開発スキルを移転・習得しながら自社開発を目指す
社外パートナー活用	自社が要件・仕様などオーナーシップを持つが，開発は外部パートナーを活用

現実的には 2 章で述べたように IT 技術者の不足は深刻化しており，内製の IT 技術者確保はなかなか進んでいない．このため，現在開発に直接携わっていない要員については，スキル習得の難易度が高いスクラッチ開発よりも，直感的な GUI 操作により業務機能を実現できる，スキル習得の難易度・コストが比較的低い技術やツールを選択することが，要員育成の効果を高め，内製化を促進できる（5.1 節 図 3 ③）．

具体的には競争領域（守り）の一部，非競争領域の一部について，ローコードを選択することが現実的である場合が多い．データ設計や一部のロジック作成のために一定の IT スキルを求められるが，ベンダーから提供される教育サービスなどを活用して IT スキルを習得することで，ある程度複雑な機能のシステムも開発できることが見込まれる．開発チームとしてローコードを選択する場合は，基本的なアーキテクチャの理解と標準化の知識も習得できるように計画するべきである．このように，どのビジネス領域に自社のリソースを注力するか，そのために要員をどう教育するのか，合わせて検討することが重要である．

一方，各部署におけるボトムアップ型の改善活動の手段としては，ローコードよりシンプルで非 IT 技術者による短期開発ができるようなノーコードが候補になる．ただし，システムの拡張性が低いなど制限も多い点，またかつての Excel マクロのように全社的な統制を効かせにくく，仕様が不明確でメンテナンスが難しいシステムが乱立してしまう可能性がある点に注意すべきである．

また，主に非競争領域において，業務に携わる担当者が自部署の業務を効率化する目的の場合は RPA の採用を検討する．自動化により担当者は別の作業に従事でき，手作業によるミスの削減や，既存システムに変更を加えることなく導入できることなどがメリットとなる．

5.5　企業レベルの標準化

最後に企業レベルの標準化として，全社横断で取り組む再利用シナリオについて説明する．

このシナリオでは，再利用時の観点として，システム開発のライフサイクル全般を通した，生産性・保守性・品質の向上をはじめ，目的に応じた様々な考慮が求められる．

競争領域において戦略的に市場に早く製品を投入することを最優先する場合は，その時点で最先端の技術を採用し，状況に応じてスクラップ＆ビルドを行うことが考えられる．しかし，通常新しい技術を個別のプロジェクト毎に検討・検証することは初期コストが重複するだけではなく，バラバラな技術が採用されることでシステムの保守コストも増大する．

企業レベルで新しい技術や手法を検討する場合は，IT 化が見込まれる事業領域，ビジネスの方向性を把握し，中長期で取り組むべき技術方針を定めていくことが望ましい．さらに全社横断でナレッジを蓄積し共有することで要員育成が効率化し，新しい技術への取り組みを加速させることができるようになる．

開発期間に入る前の最初の取り組みとして，OSS フレームワークなどを中心に，採用する技術の組み合わせを定義することにより，開発するソフトウエアの品質が安定し，保守性の向上を見込むことができる（5.1 節 図 3 ④）．さらには同じ技術の組み合わせを使うことによって IT 技術者のプロジェクト間の流動性が高まることが期待できる．

IT の統制の観点では，生産性・保守性・品質などだけではなく，ライフサイクルが中長期に渡るシステムでも利用できることを想定して，デファクトスタンダードであることや適切な技術的サポートが得られることも重要である．またベンダーロックインに陥らないためにもデファクトスタンダードな技術，可搬性の高い技術（コンテナ等）を検討すべきである．こうした技術の選定・検証から適用までのナレッジを全社レベルで共有することで，トラブル時の相互支援，再利用する部品アップデート情報やサポート情報の共有などができるようになる．

さらにシステムのモダナイゼーションを含めた開発全般の統制を推し進めるためには，共通の開発運用基盤を定め，アプリケーションアーキテクチャも含めた標準化を策定し推進することが有効である．標準化の範囲としては，クラウド基盤やアプリケーション開発ツール，部品 SaaS などを定めるとともに，非競争領域における業務 SaaS を補完する周辺開発のツールとして，ローコード開発基盤など，IT 戦略に合わせた領域をカバーすることが望ましい．

標準化のより高度な取り組みとしては，ビジネス成長に備えシステムをサービスとして再利用するマイクロサービスアーキテクチャを検討することがある．この場合サービスの粒度などの分割指針やサービス間通信やトランザクションの管理方法と障害時の対応など，多岐にわたる複雑な検討を要するため，マイクロサービス化を目指す目的を念頭に置き，手段が目的とならないように慎重に取り組むことが肝要である．

また，部品レベルの再利用手段である OSS フレームワークなどは外部から調達することが一般的だが，将来へ向けた取り組みとして共通利用するソフトウエア自体の開発を行うオープンソース・インナーソースも企業レベルの標準化に含めることができる．企業内での標準化としては，まずはインナーソースのモデルを今後検討すべき取り組みと位置付けることができる．

6. お わ り に

本稿では企業システムを構築する技術の選び方として，国内の動向や課題の整理，システム構築技術の概要と動向，利用シナリオを俯瞰的にまとめた．

具体的にはシステム構築に関する技術を俯瞰的に捉え，「効率的に作る技術」「再利用する技術」に大別し，利用シナリオと選択すべき技術の組み合わせを整理し提示することにより，

IT 戦略・人材戦略に応じた技術の選択ができるようになったと考える．

システム構築を取り巻く技術の進化のスピードは速く，新しい技術や手法も次々に現れている．開発者には最新技術を学び，自身の技能を向上することが求められるとともに，やみくもに最新技術を追うのではなく，技術を使える人材やコスト面も踏まえ，将来的にも主流であると考えられる技術を選択する「技術の目利き力」が重要となる．適切な技術の組み合わせを選び，開発プロセスとともに活用することによって，目的に沿ったシステムを適切に作ることができる．

特に AI ネイティブ開発領域は今後も飛躍的に進化し，モデル自体の能力向上，自然言語での操作の充実，目的別に最適化されたプラグインやパーツの流通などが予測され，継続的に動向把握すべき領域である．

また，IT 技術者調達の観点では，慢性的に不足する人材を育成するために，外部の技術パートナーを活用して要員育成を図ることも重要である．特に内製化を目指す場合には，要員スキルとビジネスの目標を踏まえて，ローコードや RPA などの開発ツールを適用する範囲を正しく設定することも重要なポイントとなる．

当社では，これまでシステム構築で培ってきた知見を基に，当社の「目利き」によってニュートラルな立場で最新技術をあらかじめ選択し，プロセスと最適に組み合わせた「AlesInfiny（アレスインフィニィ）」というソリューションを提供している．顧客の技術選択作業の負担を軽減し，既存システムのモダナイズ実現，アジャイルな取り組み，内製化へのアプローチなど，様々な顧客課題への取り組みを加速することを目的としている．また，整理した技術と適用シナリオは 2023 年末時点での組み合わせであるため，継続的な情報収集と試行による改善により，顧客にとって最適な技術と組み合わせを取り込み発信していく所存である．

本稿が企業のビジネス戦略に合わせたシステム構築方針の検討時に，技術の動向を俯瞰的に捉えて整理し，システムの目的に適した構築技術の組み合わせを検討するための一助となれば幸いである．

最後に，本稿の執筆にあたりご協力・ご指導いただいた関係者各位に，この場を借りて深くお礼を申し上げる．

＊ 1　System of Engagement の略．顧客とのつながりを意識した IT システム．
＊ 2　System of Insight の略．データ分析によって顧客のニーズや購買プロセスを洞察（Insight）するシステム．
＊ 3　System of Records の略．記録するためのシステム．
＊ 4　Robotic Process Automation の略．ソフトウエアロボットを使用して人間が行っていたルーチンワークを自動化する技術．

参考文献

［1］ デジタルガバナンスコード 2.0，経済産業省，2022 年 9 月，https://www.meti.go.jp/policy/it_policy/investment/dgc/dgc2.pdf
［2］ 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
［3］ 企業 IT 動向調査報告書 2023，一般社団法人 日本情報システム・ユーザー協会，2023 年 4 月，https://juas.or.jp/cms/media/2023/04/JUAS_IT2023.pdf
［4］ DX レポート 2.1，経済産業省，2021 年 8 月，https://www.meti.go.jp/press/2021/08/20210831005/20210831005-2.pdf
［5］ Andy Oram 著 /InnerSource Commons Japan 訳，インナーソース入門，2015 年 7 月，https://jp-contents.innersourcecommons.org/v/getting-started-with-innersource/

［6］ 姉崎章博，OSS ライセンスを正しく理解するための本，株式会社シーアンドアール研究所，2021 年 10 月，p.12-34.
［7］ OSI Approved Licenses, Open Source Initiative, 2023, https://opensource.org/licenses/

※　上記参考文献に挙げた URL は 2024 年 1 月 5 日時点での存在を確認.

執筆者紹介　瀬　嵐　雅　樹（Masaki Searashi）
1990 年日本ユニシス(株)入社．2004 年からアプリケーション開発技術の体系化・標準化と社内への適用・推進活動に従事．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」の適用推進活動に従事.

真　野　　悟（Satoru Mano）
2000 年日本ユニシス(株)入社．金融業界向け対外系接続システム開発，自動認識技術関連システム開発を経て，現在は共通技術部門に所属し，フロントエンド開発，モバイル，システム開発セキュリティ等の技術支援に従事.