GeneXus Day 2026

システム開発の羅針盤

－DXを加速するモダナイゼーションとローコード×AIによる進化－

2026年2月5日

シニア・アナリスト　入谷 光浩

株式会社アイ・ティ・アール

iTR

入谷 光浩
Mitsuhiro Iriya

シニア・アナリスト
ITR

【 業務内容 】

- 市場・技術動向に関する調査とレポート執筆
- ユーザー企業のDX・IT戦略コンサルティング
- ベンダーのビジネス・製品戦略支援コンサルティング
- 外部セミナーや社内研修での講演

【 専門分野 】

- クラウドサービス（IaaS、PaaS）
- インフラストラクチャ
- アプリケーション開発／DevOps
- IT運用／サービスマネジメント

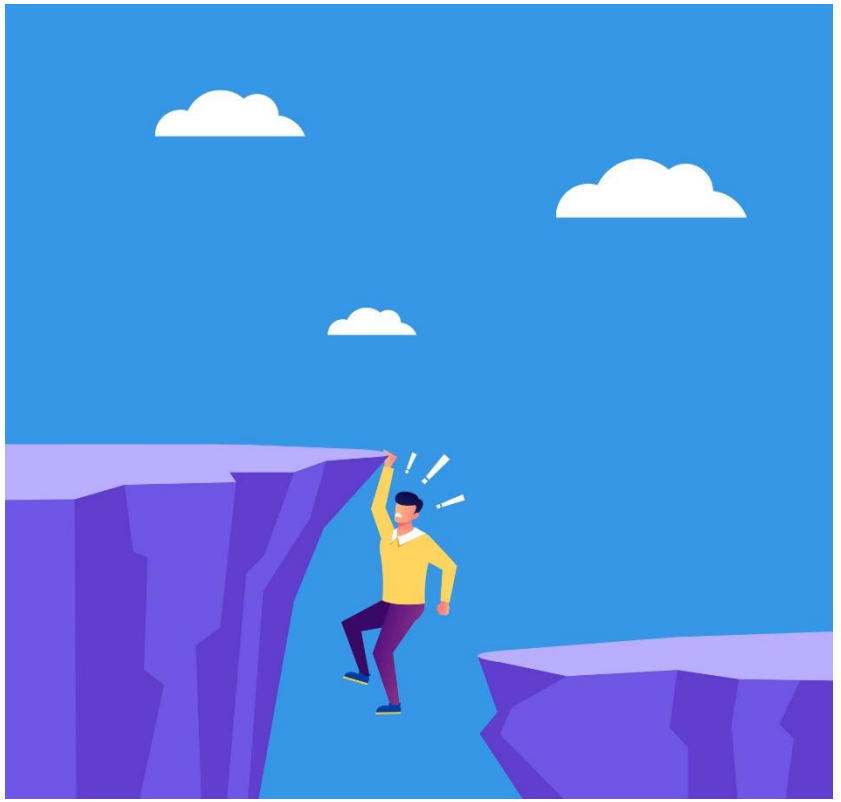
【 経歴 】

- ITアナリスト歴20年以上
- 外資系リサーチファームで15年間アナリストとして従事し
  クラウド・ソフトウェア・セキュリティの日本市場調査責任者
- 複数の外資系ITベンダーにて事業戦略・新規事業開発を担当
- セミナーでの講演実績多数
- ITmediaエンタープライズで連載中
  「新しい乱世」を生き抜くためのIT羅針盤

2025年の崖は乗り越えられたのか？

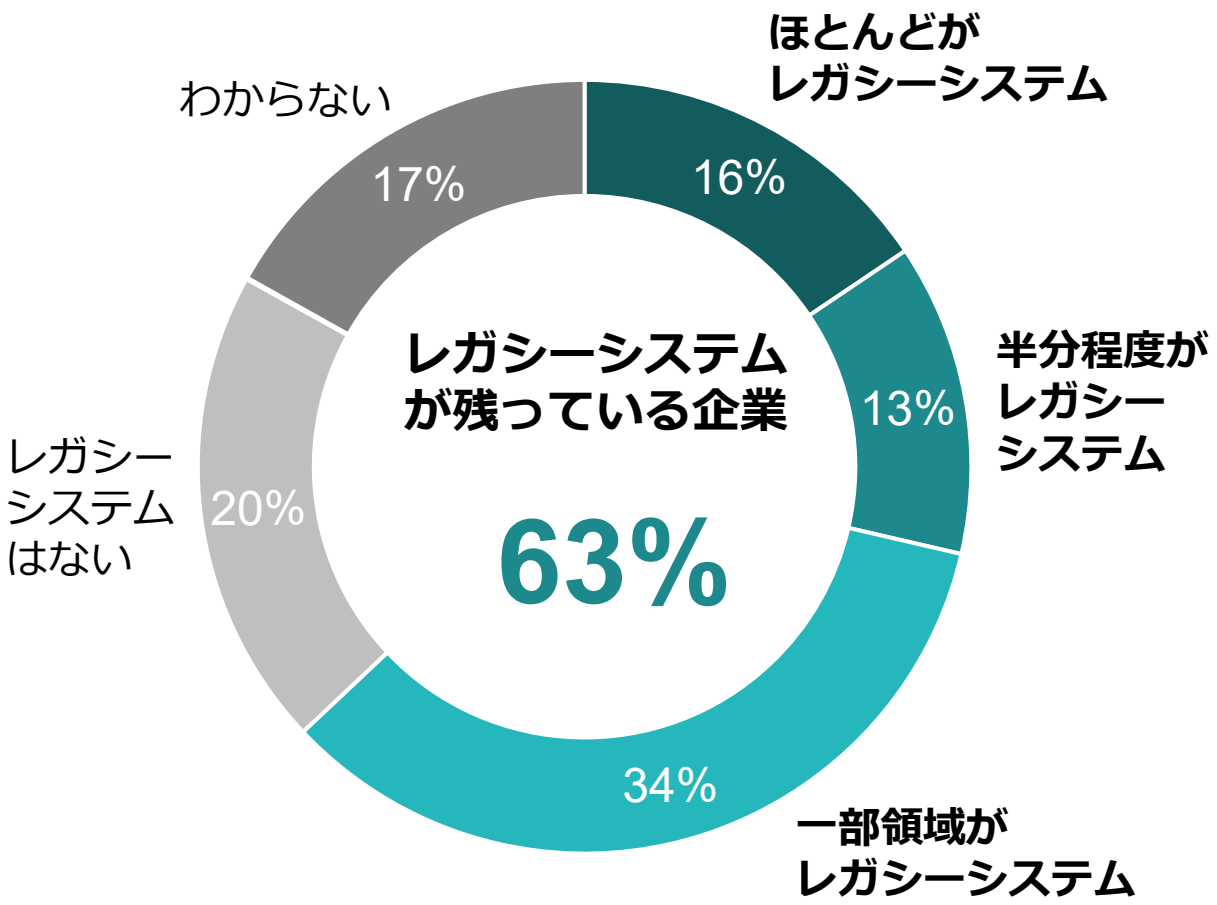
DXレポート（2018年公開）で示された「2025年の崖」の課題

- レガシーシステム（技術的負債）の継続
- 複雑化・ブラックボックス化し新技術の活用が困難
- 運用保守にかかるコストと人員の負担増加
- サイバーセキュリティや障害のリスクが高まる
- ベンダーへの委託（丸投げ）から脱却できない
- DX人材の確保ができない



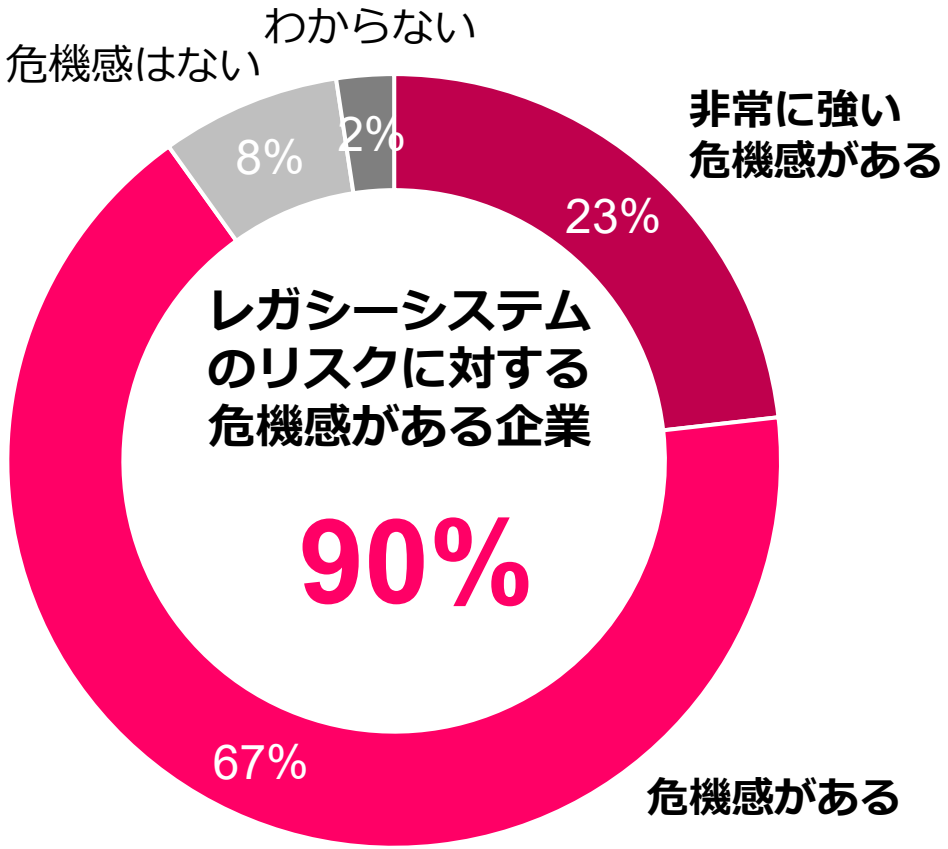
企業はこれらの課題を解決できなければDXが実現できないのみではなく
2025年以降、最大12兆円／年の経済損失が生じる可能性がある

レガシーシステムの状況



（N=1,516）

出典：IPA『DX動向2025』

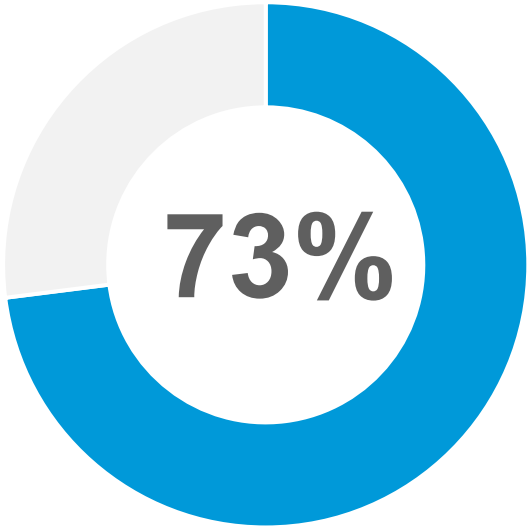


（N=305）

出典：ITR『ITインフラ動向調査2025』

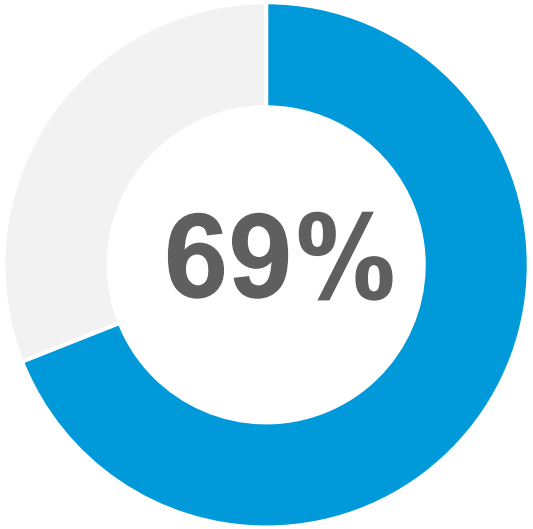
レガシーシステムの維持管理がIT部門を苦しめている

運用保守業務負担が3年間
増加し続けている企業



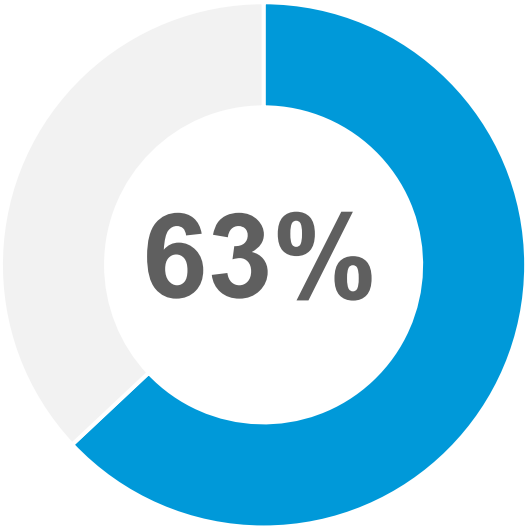
人手不足

運用保守コストが3年間
増加し続けている企業



コスト増

外部事業者に運用保守業務を
委託している企業



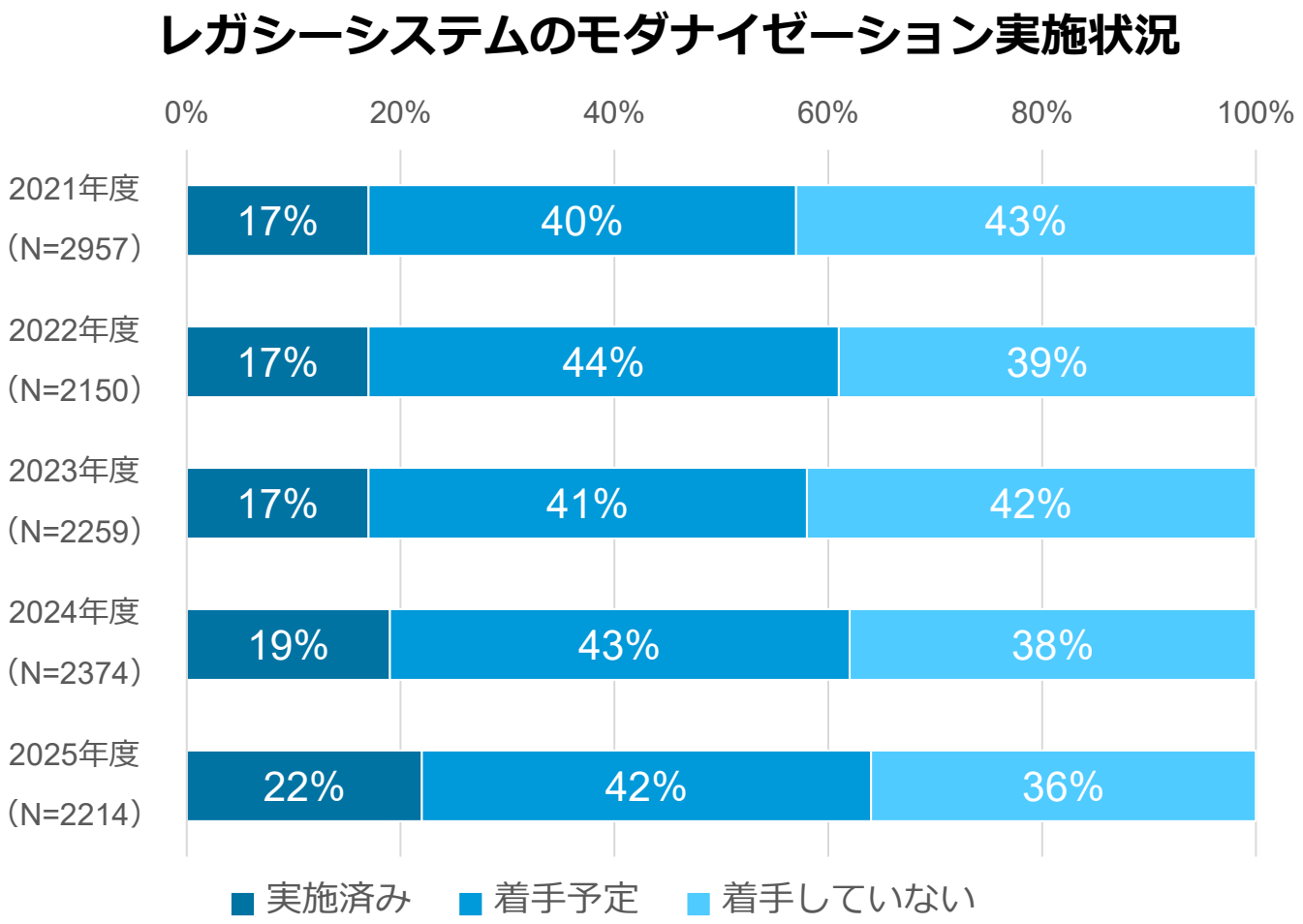
スキル空洞化

7年間で「崖」を乗り越えることはできなかった

N=333
出典：ITR『システム運用管理の実態調査2025』

なぜモダナイゼーションは進まないのか？

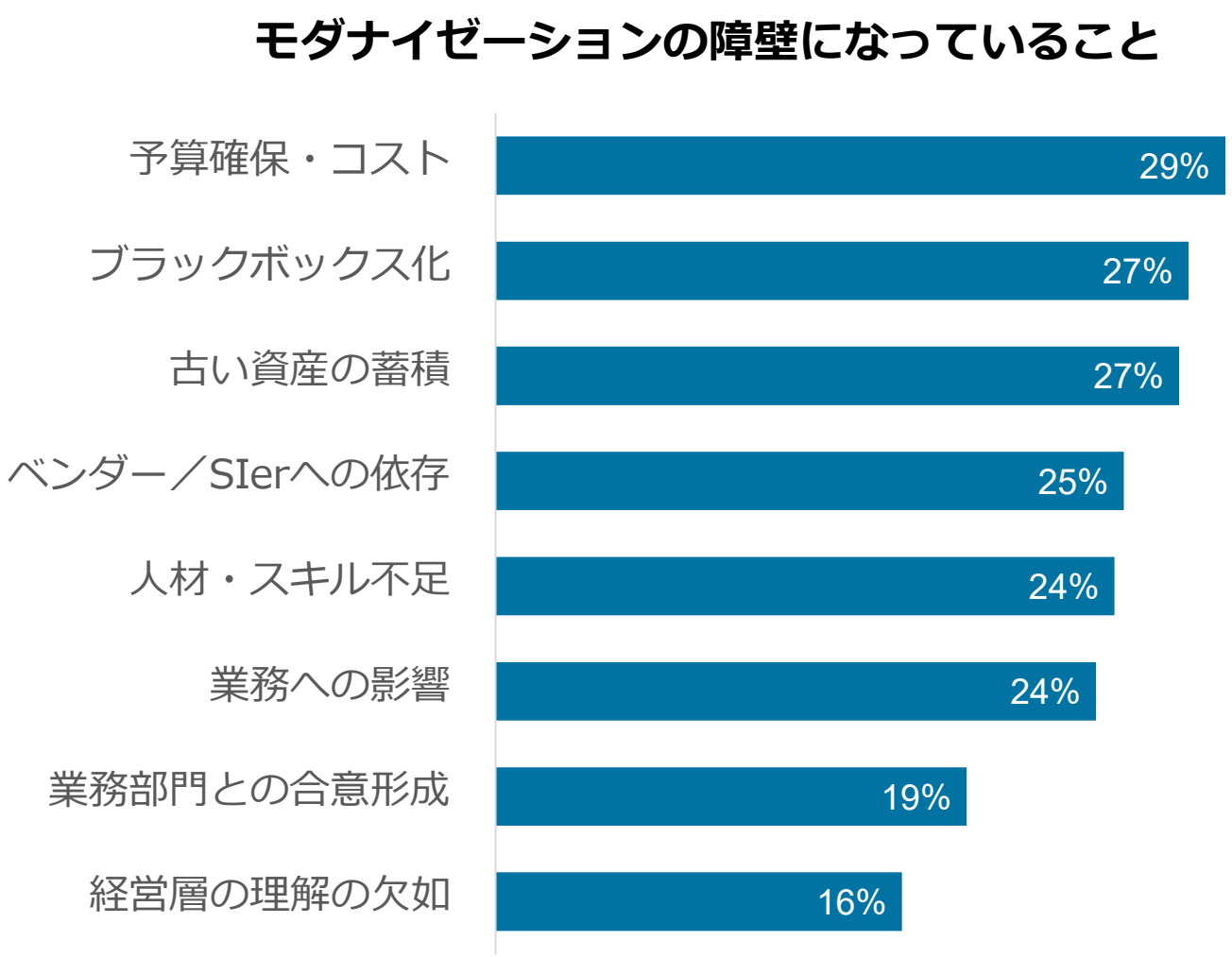
モダナイゼーションは停滞している



- 多くの企業がモダナイゼーションの必要性は認識している
- 完了している企業はまだ少なく5年間であまり増えていない
- 大半の企業が、検討段階や計画段階で足踏みをしている、あるいは着手できていないのが現状である

※2021・2022年度は「レガシーシステムの撤廃」として調査
出典：ITR『IT投資動向調査』

モダナイゼーションを阻む構造的課題



N=274

出典：ITR『ITインフラ動向調査2025』

① 現行踏襲を求める意思決定

- 既存業務の維持が最優先
- 資産がそのまま引き継がれる
- 投資対効果が見えにくくなる

② ブラックボックス化のリスク

- 手を入れることへの恐怖心
- 業務への影響範囲がわからない
- 目的が技術更新にすり替わる

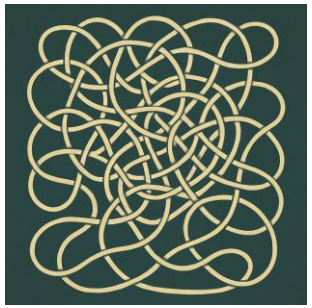
③ 人材・スキルの制約

- レガシーを理解する人材の減少
- 新技術の習得に割く余力がない
- 丸投げが染みつき主導できない

DXを加速させるモダナイゼーションに求められていること

レガシーシステムによる足かせ

- 現行業務・機能の再現
- 複雑化・ブラックボックス化
- 陳腐化技術の延命



DX推進の障壁

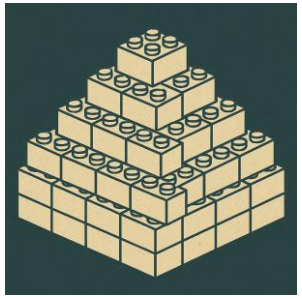
スピードの欠如

負荷・コストの増大

モダナイゼーション

モダナイズによる加速

- 価値起点の再設計
- 柔軟性・俊敏性
- 最新技術による継続的進化



DX推進の加速

不確実性への対応

ビジネス価値の創出

求められているのは「一度きりのシステム刷新」ではなく
DXを加速するために「変化し続けられる開発構造」への転換

変化を続けられる開発構造とは？
－モダナイゼーションの焦点－

開発構造は4つの仕組みで決まる

①アーキテクチャ

変化しやすい構造

分割・疎結合・API化で
変更・修正の影響を最小化する

②データ

AI活用の前提を整える

連携・品質・権限を整え
AIで使えるデータにする

③デリバリー

速く・安全なリリース

CI/CD・テスト自動化で手戻り
を減らしリリース頻度を上げる

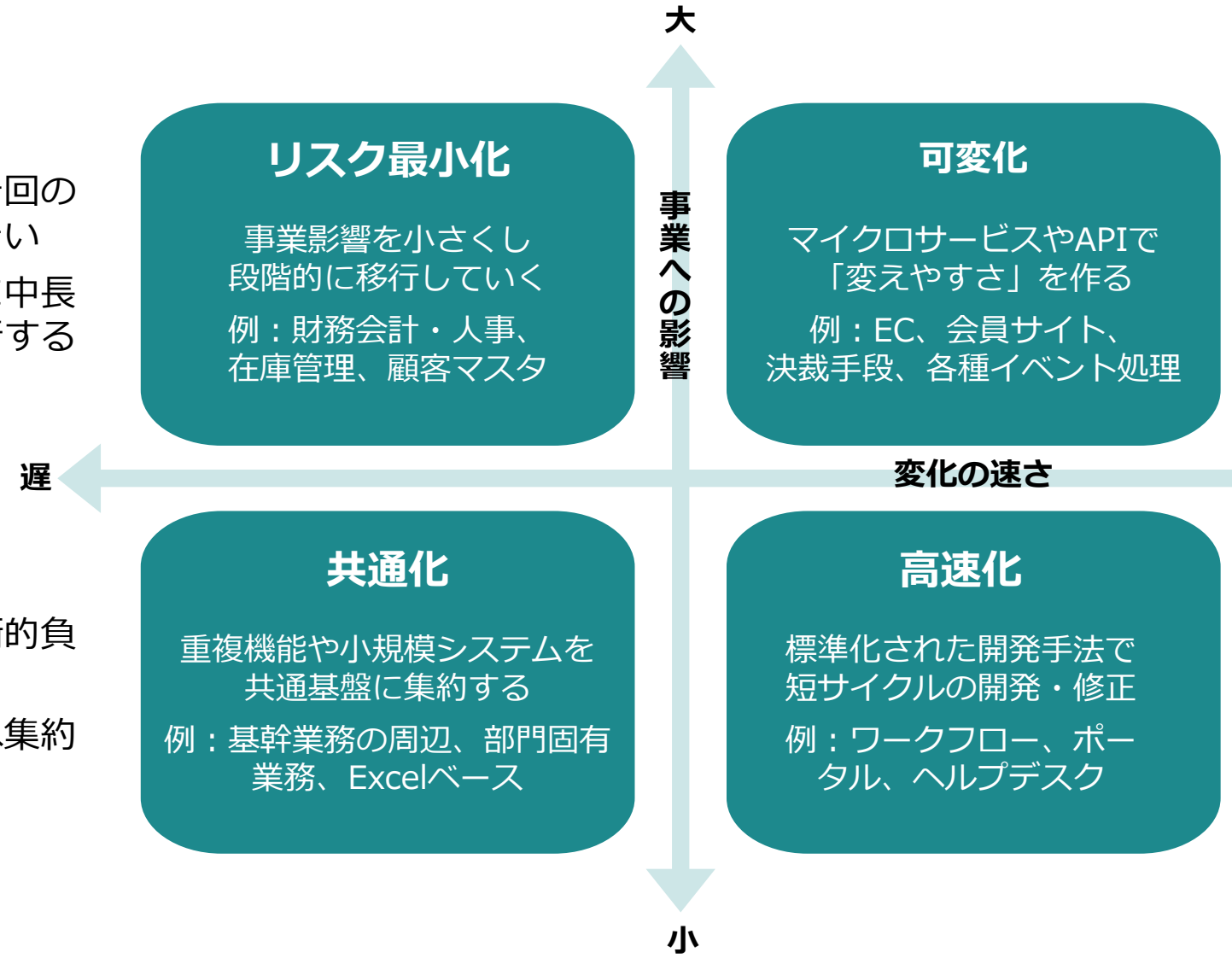
④ガバナンス

標準化とガードレール

部品・テンプレート・ルールで
速さの持続とばらつきを抑える

開発構造を変えるとは、アーキテクチャ・データ・デリバリー・ガバナンスの4要素を
セットで整え、変化を小さく・速く・安全に回せる状態に作り替えること

モダナイゼーションのタイプを分類しどこに集中するかを決める



- 変化は少ないが一回の変更の影響が大きい
- 安定稼働を第一に中長期で計画的に移行する

- サービスや顧客ニーズの変化が激しい
- クラウドネイティブ化で俊敏性を高める

- 重複や乱立は技術的負債の温床になる
- 早期に共通基盤へ集約し負債を減らす

- 直接的な影響は小さいが変更や追加が多い
- 業務自動化などDX成果が得られやすい

変化を前提にすると、従来型の開発構造では苦しくなる

- コード・業務・データが密結合し
  変更のたびに影響が読めない
- 設計・実装・テストが属人化し
  スピードが人に依存する
- 品質とスピードがトレードオフになり
  ガバナンスを強めるほど遅くなる

開発構造の変革にローコード開発が果たす役割

①アーキテクチャ

変化を持続させる

- 画面・ロジック・データの分離
- API前提の実装を可能にする

②データ

データを扱いやすくする

- 統合化されたデータモデル
- データの連携・品質を担保

③デリバリー

スピードを属人化させない

- CI/CDやテスト自動化を実行
- 変更・リリース単位を最小化

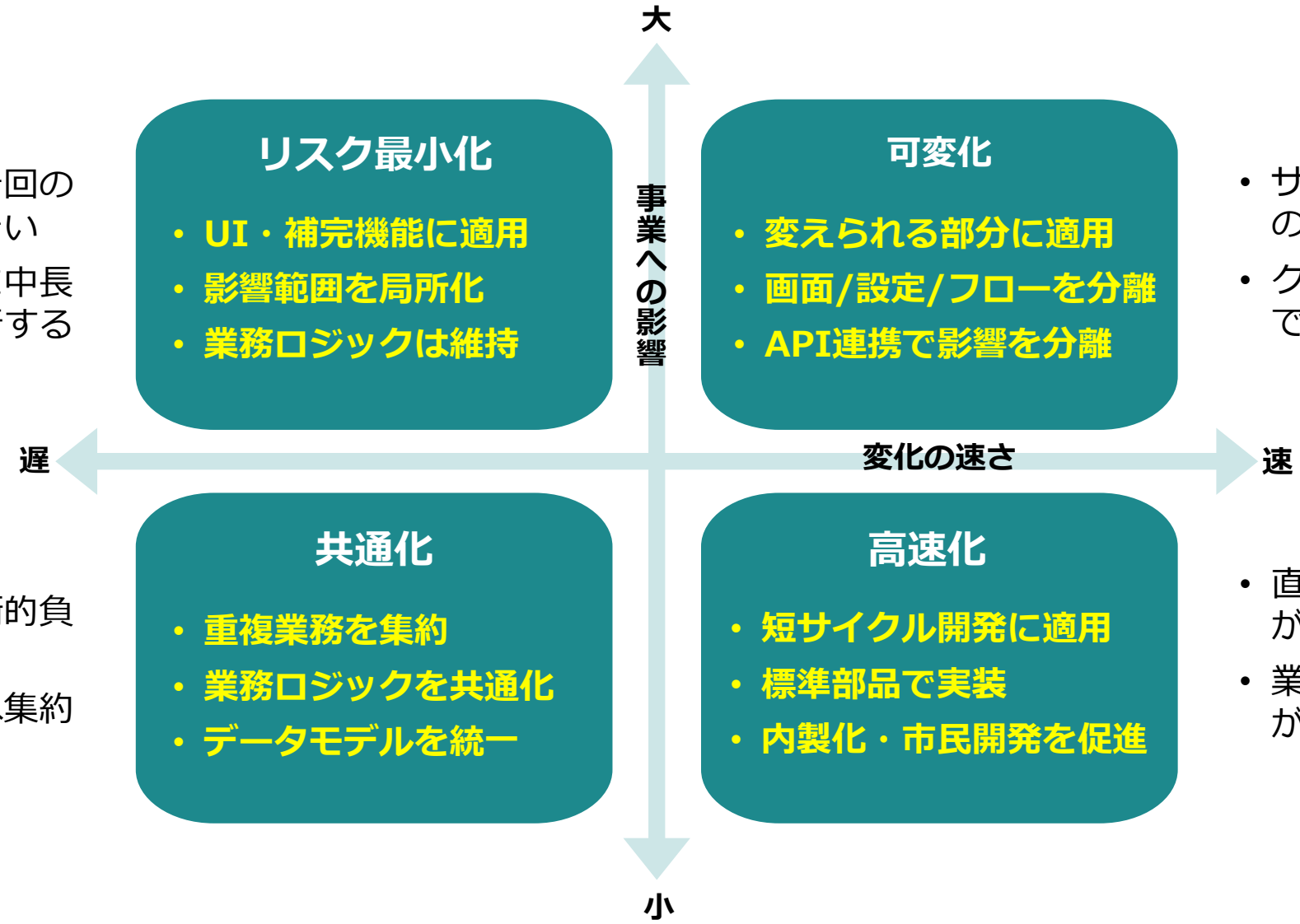
④ガバナンス

レールを敷く

- 部品やテンプレートの活用
- 権限・ルールの仕組みを内包

ローコード開発は「開発を速く簡単」にするための手法ということではなく
「変化を小さく・速く・安全に回し続ける」ための開発構造を成立させる手法

モダナイゼーションタイプ別のローコード開発適用方針の例



- 変化は少ないが一回の変更の影響が大きい
- 安定稼働を第一に中長期で計画的に移行する

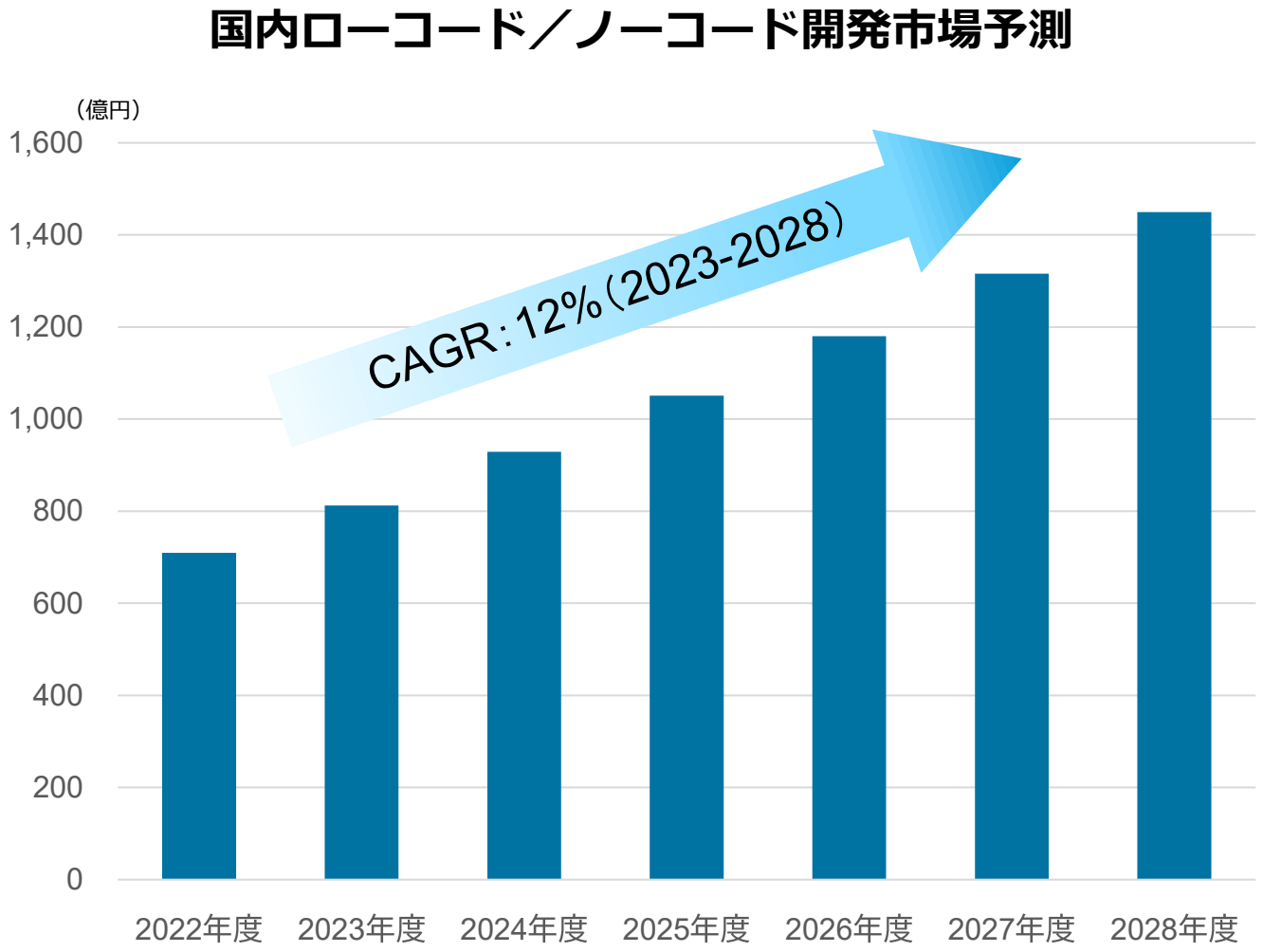
- サービスや顧客ニーズの変化が激しい
- クラウドネイティブ化で俊敏性を高める

- 重複や乱立は技術的負債の温床になる
- 早期に共通基盤へ集約し負債を減らす

- 直接的な影響は小さいが変更や追加が多い
- 業務自動化などDX成果が得られやすい

ローコード×AIによって
開発はどのように進化していくのか？

ローコード開発基盤の導入が急速に拡大している



出典：ITR『Market View：ローコード／ノーコード開発市場2025』

- 大企業から中小企業に至るまで幅広い層に導入が拡大
- 開発コスト・期間の削減効果
- モダナイゼーションへの適用
- DX・業務変革の推進
- 市民開発・内製開発の実現

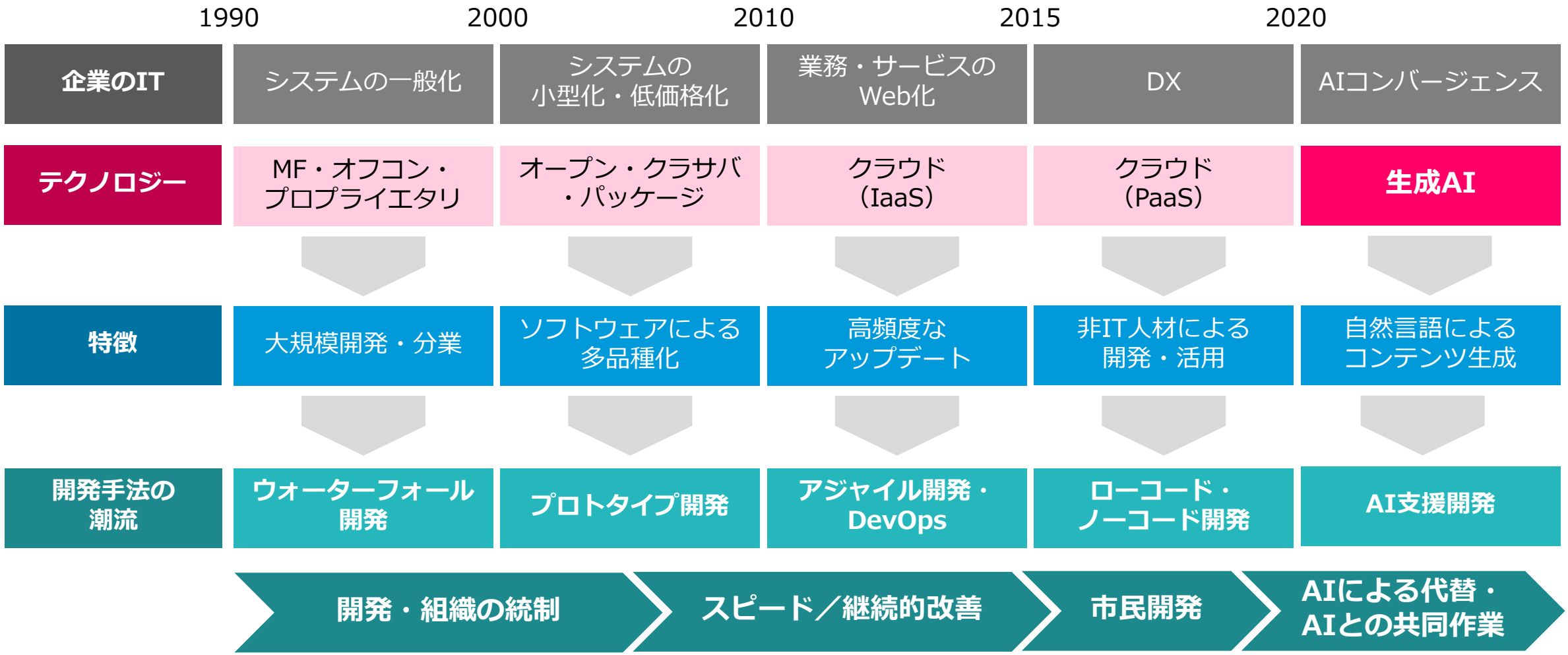
「AI×開発」は2026年度の最重点投資領域になる

2026年度に新規導入／投資増加が期待される上位10製品（全110製品中）

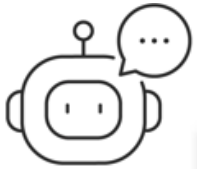
	新規導入可能性	投資増減指数
1位	AI／機械学習プラットフォーム	生成AI
2位	AI支援開発	AI／機械学習プラットフォーム
3位	生成AI	AI支援開発
4位	チャットボット／チャットサポート	音声認識
5位	VR／AR／MR	運用自動化／AIOps
6位	RPA	CDP／プライベートDMP
7位	エッジコンピューティング	BI／セルフサービスBI／ダッシュボード
8位	予算／プロジェクト管理	DevOps
9位	iPaaS	セールスイネーブルメント
10位	PaaS	脆弱性診断／レッドチーム演習

出典：ITR『IT投資動向調査2026』

ソフトウェア開発技術の変遷



AIが効果を発揮できる開発構造への転換を図る

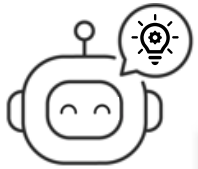


AIが効果を発揮しにくい開発構造

- 仕様や設計がバラバラ・暗黙知化
- 業務ロジック/画面/データが密結合
- 変更の影響範囲が人の経験に依存
- 設計・実装・テストが工程ごとに分断

その結果↓

- ➢ AIが要件や設計を正しく理解できない
- ➢ AIの妥当性を人間がその都度検証
- ➢ 実用化するためには人間の支援が必要



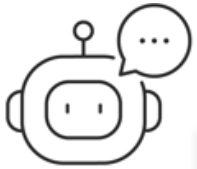
AIが効果を発揮しやすい開発構造

- 要件・業務ルールが構造化されている
- 業務ロジック/画面/データが分離
- APIとモデルが定義され影響範囲が明確
- 標準化された設計ルールや部品がある

その結果↓

- ➢ **AIが構造に基づき各工程を横断的に支援**
- ➢ **AIが影響範囲・修正候補・テストを提示**
- ➢ **人間は意図提示・レビュー・判断に専念**

AIが効果を発揮できる開発構造への転換を図る

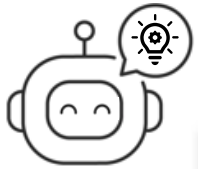


AIが効果を発揮しにくい開発構造

- 仕様や設計がバラバラ・暗黙知化
- 業務ロジック/画面/データが密結合
- 変更の影響範囲が人の経験に依存
- 設計・実装・テストが工程ごとに分断

その結果↓

- AIが要件や設計を正しく理解できない
- AIの妥当性を人間がその都度検証
- 実用化するためには人間の支援が必要



AIが効果を発揮しやすい開発構造

- 要件・業務ルールが構造化されている
- 業務ロジック/画面/データが分離
- APIとモデルが定義され影響範囲が明確
- 標準化された設計ルールや部品がある

ローコード開発

その結果↓

- **AIが構造に基づき各工程を横断的に支援**
- **AIが影響範囲・修正候補・テストを提示**
- **人間は意図提示・レビュー・判断に専念**

- AI支援開発のボトルネックはAIの精度ではなく開発構造の曖昧さにある
- ローコード開発基盤はAIが迷わず働ける開発構造に整えていく役割を担う

ローコード開発ベンダーのAI対応が進む

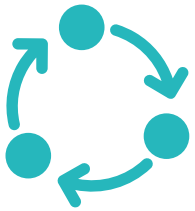
- ローコード基盤とAIとの統合を積極的に推進
- 自然言語による指示からアプリケーション設計や業務ロジックを自動生成する機能の強化が進んでいる
- 画面・データモデル生成、コーディング支援、テスト生成・自動化など各タスクでのAI対応も進んでいる
- 開発スピードの飛躍的な向上やエンジニア負荷の大幅な軽減が期待される
- 市民開発や内製開発をさらに加速させる

AIを実装したローコード開発は、従来の補助的な開発手段から、IT戦略の中心的な開発手段へと位置づけが変わっていく

ローコード×AIはビジネス価値創出に向けてどのように貢献できるのか？

価値創出のスピードと質を同時に高める

要件定義から設計・実装・テストまでをAIが横断的に支援することで
短期間でのリリースと安定した品質を両立できる



人材の制約を越えて、継続的な開発とリリースを可能にする

タスクをローコード基盤とAIになるべく任せ、人間は企画や判断に
集中できるため、限られた人員で開発サイクルを回し続けながら
新しいアプリケーションやサービスを継続的にリリースできる

将来に向けた開発構造の資産を蓄積していく

業務やデータのモデル、開発に必要な部品が基盤上に構造資産として残り
AIが次の開発や変更対応に再利用しながらアップデートしていくことで
創出する価値が継続的に高まっていく

これからのシステム開発は
変化が起きるたびに作り直す開発ではなく
変化を内包できる開発構造にしていくべきである

ローコードとAIは
その構造への転換を図るための最適な手段となる

問いを、答えに。