スマラボ
Smart Labo

WHITE PAPER / 2026

AI駆動開発 × 既存保守の再定義

― 「日・越・AI」の三者分業モデル完全ガイド ―

Presented by
IDS Corporation

内容

	はじめに	2
第1章	日本のIT開発が直面する構造的限界	3
第2章	なぜ従来のオフショアでは解決できなかったのか	5
第3章	AIが開発構造を変える決定的な理由	7
第4章	AIによって再定義される「既存保守」という領域	9
第5章	新規開発と既存保守を分離する分業モデル	11
第6章	この分業モデルにおいて、なぜベトナムなのか	13
第7章	AI時代に最適な契約形態	15
第8章	それでも失敗する企業の共通点	16
第9章	2026年型・三者分業モデルの全体像	18
	おわりに	19

はじめに

本ホワイトペーパーは、2026年に日本企業のIT開発が直面している「構造的限界」を出発点に、なぜ今、AIとオフショアを切り離さず一体の仕組みとして再設計する必要があるのか整理することを目的としています。

近年、多くの企業がDXや内製化を掲げAIやクラウドといった新技術を積極的に導入してきましたが、ツールや技術は進化しているにもかかわらず開発現場の負荷は軽減されていません。その背景には技術以前の「構造」の問題が存在しています。

また、経済産業省の調査によると日本のIT人材は2030年までに最大累計79万人が不足すると予測されており、これには単なる人数の不足ではなく、限られた人材が多様な業務を同時に抱え、本来じっくり考えるべき事案に時間を割けず、目の前の作業に追われ続けている状態が常態化しているのです。

本稿では現在の構造から出発し、従来手法の限界、AIによる構造変化、それを前提とした分業の再設計、そして実行可能なモデルへと至る流れを、9章構成で整理しています。読者の皆様には、「なぜこの構造が必要なのか」という問いを共有していただきたいと考えています。

第1章｜日本のIT開発が直面する構造的限界

2026年現在、日本のIT開発現場では、すでに限界が顕在化しています。慢性的なエンジニア不足は長年指摘されてきましたが、問題は単なる人数の不足ではありません。新規開発、既存システムの保守・運用、障害対応、問い合わせ対応といった多様な業務が、限られた人材に集中し、思考と作業が常に奪い合われている点に本質があります。

1.1 思考と作業に奪われ続ける時間

典型的な開発現場では、以下のような状況が日常的に発生しています。午前中は新規機能の設計会議に参加し、午後は既存システムの障害対応に追われ、夕方はチームからの問い合わせに対応する。そして夜間にようやく、本来の開発作業に着手できる。このような状況では、深い思考を要する設計や、複雑な問題の解決に必要な集中力が、常に中断されるリスクに晒されています。

人間の認知リソースは有限です。マルチタスキングが効率的でないことは、認知科学の研究で繰り返し示されてきました。しかし、多くの開発現場では、この科学的知見が無視され、一人のエンジニアに複数の役割が同時に課せられています。結果として、どの業務も中途半端になり、品質が低下し、さらに問題が増えるという悪循環が生まれています。

1.2 経験豊富な人材の疲弊と離脱

一方で、経験豊富な人材ほど業務負荷が高まります。複雑な問題の解決、意思決定、若手の育成、クライアント対応といった高度な業務が経験豊富な人材に集中し、技術的な興味や成長意欲を持ち続けている優秀なエンジニアが、管理業務や雑務に追われ本来の技術的な仕事に集中できない状況に陥ります。このような状況が続くと彼らはより技術的なチャレンジを求めて、スタートアップや外資系企業に転職するか、あるいはフリーランスとして独立することを選択し、結果として、離職や現場離れが進み、組織は最も価値の高い人材を失い続けていくことになります。

1.3 若手育成の構造的困難

若手は育つ前に現場に投入され、十分な経験を積む前に疲弊していきます。本来であれば、先輩エンジニアから設計思想や判断基準を学び、段階的に責任の範囲を広げていくべきです。

入社1年目のエンジニアが十分なレビューも受けられないまま、本番環境に影響するコードを書くことを求められるケースが少なくありません。レビューする側も時間がなく、表面的な指摘にとどまり、設計思想や判断の背景を伝える余裕がない。このような環境では、若手は技術的なスキルは身につけても、なぜその判断をしたのか、どのような前提で設計したのかを理解する機会を失います。

1.4 DXと内製化の落とし穴

DXや内製化は一見前進しているように見えますが、実態としては「やることが増えただけ」というケースも少なくありません。新しい技術を導入しても、役割分担や意思決定の構造が変わらなければ、負荷は依然として人に集中します。

例えば、クラウドネイティブなアーキテクチャを導入した企業でも、運用や監視の責任が明確に分離されず、開発チームが24時間体制で対応を求められるケースがあります。マイクロサービス化を進めたものの、各サービスの責任範囲が曖昧で、障害時の切り分けに時間がかかる。このように、技術は進化しても、組織の構造や役割分担が変わらなければ、問題は解決しません。

1.5 構造的限界の本質

ここに、日本のIT開発が抱える構造的な限界があります。問題は、個々のエンジニアの能力や努力の不足ではありません。システムの複雑化、業務の多様化、人材の不足が同時に進行する中で、限られた人材にすべての責任が集中する構造が、長期的に組織の開発力を削ぎ続けているのです。

この構造的な詰まりを解決するには、技術的な改善だけでは不十分です。開発の進め方、役割分担、情報の流れ、意思決定のプロセスといった、開発の「構造」そのものを再設計する必要があります。

第2章｜なぜ従来のオフショアでは解決できなかったのか

人材不足を補う手段としてオフショア開発は広く活用されてきましたが、実際には十分な効果を実感できていない企業も少なくありません。コミュニケーションの難しさ、品質のばらつき、日本側の管理負荷の増大といった課題が繰り返し指摘されています。

これらの課題は、国や人材の質の問題として語られがちですが、本質はそこではありません。暗黙知に依存した進め方、曖昧な仕様、口頭や担当者の記憶に頼った判断。こうした前提を変えないまま、開発場所だけを海外に移したことが、問題を拡大させてきました。

2.1 コミュニケーションの難しさの正体

ある企業では、オフショアチームに対して「ユーザビリティを向上させてほしい」という曖昧な指示を出しました。日本側の担当者は、過去の経験や文脈から、どのような改善を期待しているかを理解していましたが、それを明文化していませんでした。オフショアチームは、自分たちの理解に基づいて改善を実施しましたが、日本側の期待とは大きく異なるものでした。

このような問題は、言語の違いや文化の違いとして説明されがちです。しかし、実際には、同じ日本人同士でも、十分な文脈を共有していない場合には、同様の問題が発生します。問題の本質は、情報の構造化と共有の仕組みが欠けていることにあるのです。

2.2 仕様書の限界

多くの企業では、オフショア開発を開始する際に、詳細な仕様書を作成します。しかし、仕様書だけでは伝わらない情報が大量に存在します。なぜその機能が必要なのか、どのような制約があるのか、過去にどのような問題があったのか、将来どのような拡張を想定しているのか。こうした情報は、仕様書には記載されず、担当者の頭の中に留まります。

さらに、仕様書は作成時点での理解を反映したものです。開発が進むにつれて、理解が深まり、前提が変わり、優先順位が変わります。しかし、仕様書の更新は後回しにされがちで、結果として仕様書と実装の間に乖離が生じます。オフショアチームは、古い仕様書を参照しながら開発を進めることになり、日本側の期待とずれが生じます。

2.3 管理負荷の増大

オフショア開発を導入した多くの企業が直面する問題が、管理負荷の増大です。日本側の担当者は、オフショアチームの進捗を把握し、品質を確認し、問題があれば対応する必要があります。しかし、オフショアチームの作業内容を理解するには、相当な時間と労力が必要です。

特に、既存システムの保守や改修をオフショアに委託する場合、日本側の担当者は、システムの全体像や設計思想を説明する必要があります。この説明には膨大な時間がかかり、結局日本側の担当者が直接対応した方が早いという状況に陥ります。よって、オフショア開発を導入したにもかかわらず、日本側の負荷が減らない、あるいはむしろ増えるという逆説的な状況が生まれます。

2.4 品質のばらつきと属人化

オフショア開発では、品質のばらつきも大きな課題となります。同じ仕様書を渡しても、担当者によって品質が大きく異なる。これは、オフショア側の人材の能力の問題として説明されがちですが、実際には、判断基準や品質基準が明文化されていないことが原因です。

日本側のエンジニアは、長年の経験から、どのようなコードが良いコードか、どのような設計が適切かを直感的に理解しています。しかし、この直感的な理解は、オフショアチームには伝わりません。日本の新人教育にも同様のことがいえます。結果として、レビュー時に大量の指摘が発生し、修正に時間がかかり、オフショア開発の効率性が損なわれます。

上記の点から、オフショア自体が機能しなかったのではなく、構造を変えないまま場所だけをオフショアに移したことが、失敗の本質的な原因であったといえます。開発の進め方、情報の共有方法、判断基準の明文化、役割分担といった構造を変えずに、開発場所だけを海外に移しても、問題は解決しません。むしろ、距離や言語の壁が、既存の問題をより顕在化させたのです。

この反省を踏まえなければ、AIを導入しても同じ失敗を繰り返すことになります。AIは強力なツールですが、構造を変えずにAIだけを導入しても、既存の問題が自動的に解決されるわけではありません。

第3章｜AIが開発構造を変える決定的な理由

前章で触れた構造を変えていく必要があることを理解して初めて、AIが本質的な意味を持ちます。AI駆動開発の価値は、コードを書く速度を上げることや、単純作業を自動化することではありません。多くの議論がこのレベルで止まっていますが、それではAIの本来の強みを生かすことができません。

3.1 情報と判断の構造化

AIが開発構造を変える決定的な理由は、「情報」と「判断」を構造化できる点にあります。従来の開発では、要件定義の背景、仕様変更の理由、設計時の迷い、妥協のポイントといった重要な情報が、人の頭の中や会議の空気感として消費されてきました。議事録に残らず、コードにも表現されず、担当者が変われば失われる。これが属人化の正体です。

例えば、ある機能の実装において「パフォーマンスを優先するため、この部分はキャッシュを使用する」という判断がなされたとします。従来の開発では、この判断の背景や理由は、担当者の記憶や、もしくはコメントとしてコードに残される程度でした。しかし、この情報が失われれば、後から見た人は、なぜこの実装になっているのかを理解できません。

AIを前提とした開発では、これらの情報をすべてテキストとして残すことが可能になります。AIがこれらの情報を横断的に解析し、関連する情報を関連付け、矛盾や曖昧さを検出し要件定義の議論、設計意図、仕様変更の経緯を蓄積することで、情報は単なる記録ではなく再利用可能な知識へと変わります。

3.2 曖昧さの可視化

曖昧な表現や矛盾は、AIによって可視化され、人が判断すべき論点として整理されます。従来の開発では、曖昧な仕様や矛盾した要件が、開発の後半になって初めて発覚することが少なくありませんでしたが、AIは、開発の初期段階からこれらの問題を検出し明確化を促すことができます。

例えば、要件定義の段階で、「ユーザーが快適に使えるように」という曖昧な表現があったとします。AIはこの表現を検出し、「快適とは具体的にどのような状態を指すのか、どのような指標で測定するのか」という問いを提示します。これにより、曖昧な要件が、具体的で測定可能な要件へと変換されます。

3.3 コミュニケーションの質の変化

この変化は、コミュニケーションの質そのものを変えます。これまで問題視されてきた言語差や文化差は、感覚論ではなく前提条件として整理されます。重要なのは、誰が言ったかではなく、何が前提で、どこが未確定なのかです。AIはこの前提整理を支援する役割を果たします。

会議やチャットでの議論を、AIがリアルタイムで解析し、前提条件、決定事項、未確定事項、次のアクションを自動的に整理することで、会議の内容が構造化され後から参照しやすくなります。

3.4 理解の速度の統一

さらに重要なのは、AIが理解の速度を揃える点です。従来、新しくプロジェクトに参加したメンバーは、数週間から数ヶ月かけてシステムの全体像や設計思想を理解していました。しかし、AIがプロジェクトの歴史や設計の背景、過去の判断、現在の課題など構造化された情報を提供することで、この時間を大幅に短縮できます。

結果として、オフショア開発の弱点とされてきたコミュニケーションの問題は、個人能力の問題ではなく、構造として制御可能な課題へと変わります。AIは魔法ではありませんが、構造を揃えることで、人と人の協業を現実的なものにします。

重要なのは、AIがすべてを自動的に解決するわけではないということです。最終的な判断は、依然として人間が行う必要があります。しかし、AIが情報を構造化し、理解を支援し、判断を促進するツールとして活用することで、情報は蓄積、構造化され誰でもアクセス可能になります。これにより開発の属人化が解消され、組織としての開発力が向上します。

第4章｜AIによって再定義される「既存保守」という領域

構造化の効果が最も大きく、かつ多くの企業で見過ごされてきたのが、既存システムの保守・運用という領域です。日本企業のIT投資の多くは、実際には新規開発ではなく、既存システムの維持・延命・改修に費やされています。しかしこの領域は長年、「仕方がないもの」「誰かが我慢して回すもの」として扱われてきました。

4.1 既存保守の現状

多くの現場では、仕様書が存在しない、あるいは存在していても現状と乖離しているシステムが稼働しています。過去の担当者の記憶や、断片的なドキュメント、コメントのないコードに頼らざるを得ず、影響範囲が分からないために改修が後回しになる。その結果、さらにブラックボックス化が進み、対応できる人材が限られていくという悪循環が生まれています。

具体例として、ある企業の基幹システムは20年以上前に開発され、その後、複数の担当者が改修を重ねてきました。しかし、仕様書は作成当時のもので、現在のシステムとは大きく異なっています。コードにはコメントが少なく、なぜそのような実装になっているのかが分かりません。過去の障害履歴や改修履歴も、担当者の記憶や、断片的なメールにしか残っていません。

このような状況では、新しい機能を追加することも、既存の機能を改修することも極めて困難です。影響範囲が分からないため慎重になりすぎて開発が進まない、あるいは影響範囲を誤って重大な障害を引き起こすことになります。

4.2 AIによる既存保守の構造化

AIはこの既存保守の前提を大きく変えます。人が時間をかけて理解していた内容を、AIが短時間でコードやログ、過去の障害履歴、チケット、設計メモといった散在する情報を横断的に解析し、システム構造や依存関係を可視化、仕様や影響範囲として提示できるようになることで、既存システムの理解が大幅に加速します。

- コード解析: コードの構造、依存関係、データフローを解析し、システムの全体像を可視化する
- ログ解析: 過去のログを解析し、システムの動作パターンや異常パターンを特定する
- 障害履歴の分析: 過去の障害履歴を分析し、脆弱な箇所や改善すべき点を特定する
- チケット分析: 過去のチケットを分析し、よく発生する問題や改善の傾向を把握する
- 設計メモの抽出: コードやドキュメントから、設計意図や判断の背景を抽出する

4.3 理解の土台整理と影響範囲の可視化

重要なのはAIが正解を出すことではなく、理解の土台を揃え、判断可能な状態を作ることに価値があります。これにより、既存保守は属人的な熟練作業から、再現性のある知識作業へと変わります。

従来の既存保守では、経験豊富なエンジニアが、長年の経験と直感に基づいて、システムの動作を理解し、問題を解決していました。しかし、この方法ではそのエンジニアがいなくなれば理解が失われ、新しいメンバーが参加する際には、同じように長い時間をかけて理解する必要があります。AIを活用した既存保守では、理解のプロセスが構造化されるため、新しいメンバーもAIが整理した情報を参照することで、理解に要する時間を大幅に短縮することが可能になります。

また、既存システムを改修する際最も困難なのは、変更がどのような影響を及ぼすかを予測することです。影響範囲の可視化においても、AIはコードの依存関係を自動的に解析し、変更が及ぼす影響を可視化します。ある関数を変更した場合、その関数を呼び出している箇所、その関数が依存しているデータ、そのデータを使用している他の機能を自動的に特定します。これにより、影響範囲が明確になり、テストの範囲やリリース計画を適切に立てることができます。

既存保守は、もはや「仕方がないもの」ではなく、構造化された情報を前提に、役割を分離し、分業できる領域へと変化します。

第5章｜新規開発と既存保守を分離する分業モデル

従来の開発体制では、新規開発と既存保守が同じ人材で行われてきました。AIによって既存保守が分業可能な業務になると、新規開発と同じ人材で抱え続ける必然性はなくなり、日本側は、事業アイデアの創出、プロダクト設計、意思決定、仮説検証といった高度な判断を要する領域に集中できます。

5.1 日本側の役割：創造と意思決定

事業アイデアの創出、プロダクト設計、意思決定、仮説検証といった高度な判断を要する領域では、市場理解、顧客理解、事業理解が重要であり、創造性と判断力が求められるため、日本側のエンジニアが本来の価値を発揮することが可能です。

日本側の役割

- 事業戦略の策定: 市場の動向を分析し、事業の方向性を決定する
- プロダクト設計: 顧客のニーズを理解し、プロダクトの機能やUI/UXを設計する
- 意思決定: 開発の優先順位や、技術的な選択を決定する
- 仮説検証: 新しい機能やサービスの効果を検証し、改善する

5.2 オフショアとAIの役割：構築と安定化

一方で、オフショアとAIは、実装、解析、リファクタリング、定型的な改善といった領域を担います。これは仕事の切り捨てではなく、価値を最大化するための役割再配置です。具体的には、以下のような業務がオフショアとAIの役割となります。

- 実装: 日本側が設計した機能を、実際のコードとして実装する
- 解析: 既存システムの構造や動作を解析し、理解の土台を提供する
- リファクタリング: 既存コードを改善し、保守性を向上させる
- 定型的な改善: パフォーマンスの改善、セキュリティの強化など、定型的な改善を行う

これらの業務は、構造化された情報を前提に再現性のある方法で実施することができます。AIが提供する理解の土台の上で、オフショアチームが効率的に作業を進めることができます。

5.3 分離による効果

新規開発と既存保守を分離することで、日本側のエンジニアは創造的な業務に集中し、深い思考を要する作業に時間を割くことができ、それぞれの領域に専門性を持った人材を配置することで、品質と効率が向上、結果として組織全体の開発力が向上します。また、オフショアチームを拡張することで、開発規模に応じて柔軟に体制を調整することが可能です。

この分離を実現するためには、AIによる構造化が不可欠です。既存保守が構造化され理解の土台が揃うことで、オフショアチームが独立して作業を進めることができ、新規開発においても、AIが提供する情報によりオフショアチームが日本側の意図を理解しやすくなります。

第6章｜この分業モデルにおいて、なぜベトナムなのか

分業モデルを前提とした場合、継続的に学習し、変化に適応できる人材の母数と環境が重要です。その観点で、ベトナムは若い人材層の厚み、国家としてのICT戦略、日本向け開発の成熟度を併せ持っています。

6.1 若い人材層の厚み

ベトナムは、人口の約60%が30歳未満という、非常に若い人口構成を持っています。これは、IT人材の供給において大きな強みとなります。毎年、多くの若者がIT教育を受け、エンジニアとしてのキャリアをスタートさせています。

また、ベトナムの若者は、学習意欲が高く、新しい技術に対する適応力が高いという特徴があります。AIやクラウドといった新しい技術を積極的に学び、実践に活かす姿勢が強く見られ、AI時代の開発において非常に重要な要素です。

6.2 国家としてのICT戦略

ベトナム政府は、ICT産業を国家戦略として位置づけ、積極的に支援しています。教育制度の整備、インフラの整備、外国企業の誘致など、様々な施策を実施しており、ベトナムのIT人材の質と量は、継続的に向上しています。

また、AIやデジタルトランスフォーメーションについても重要な政策課題として位置づけており、AI人材の育成にも力を入れています。これは、ベトナムがAI時代の開発に対応できる人材を継続的に供給できることを意味します。

6.3 日本向け開発の成熟度

ベトナムは、長年にわたって日本企業向けのオフショア開発を実施しており、日本企業の開発文化や品質基準に対する理解が深まっています。日本語を話せるエンジニアも多く、コミュニケーションの障壁が低いという利点もあり、日本のビジネス文化や開発プロセスに適応しており、日本側の期待を理解し、それに応える能力を持っています。これは、分業モデルを成功させる上で、非常に重要な要素です。

6.4 地政学的・経済安全保障の観点

加えて、地政学的・経済安全保障の観点からも、2026年時点で現実的かつ持続可能な選択肢と言えます。ベトナムは、政治的にも安定しており、日本との関係も良好です。また、知的財産権の保護や、データの取り扱いに関する法整備も進んでいます。

AIを使いこなせるかどうかは、個人の能力だけでなく、国全体の学習力にも依存します。ベトナムは、国家としてAI人材の育成に取り組んでおり、長期的に安定した人材供給が期待できます。

上記から、ベトナムは、AI時代の開発に対応できる人材を継続的に供給でき、日本企業の開発文化を理解し、長期的なパートナーシップを構築できるAI時代に適した国といえます。

第7章｜AI時代に最適な契約形態

AIと分業モデルを成立させる上で、見落とされがちだが極めて重要なのが契約形態です。多くの企業では、依然として請負型契約を前提にオフショアや外部パートナーを活用しています。しかし、この契約モデルはAI時代の開発構造と本質的に相性が良くありません。

7.1 請負型契約の限界

請負型契約は、仕様を固定し、成果物で評価することを前提としています。しかし、AI駆動開発では、理解が進むにつれて仕様や優先順位が変化することが前提となります。また、AIが蓄積した知識や文脈は、短期契約や人の入れ替えによって簡単に失われてしまいます。

- 仕様の固定化: 請負型契約では、仕様を事前に確定する必要がありますが、AI駆動開発では、理解が進むにつれて仕様が変化します
- 成果物の評価: 請負型契約では、成果物で評価されますが、AIが蓄積した知識や文脈は、成果物として評価されません
- 短期契約の問題: 短期契約では、AIが蓄積した知識が、契約終了とともに失われます

7.2 ラボ型契約（準委任）の特徴

このギャップを埋めるのが、ラボ型契約（準委任）です。ラボ型契約では、成果物ではなく、チームと稼働そのものに価値を置きます。これにより、仕様の変化や試行錯誤を前提とした進め方が可能になります。

- チームへの投資: 成果物ではなく、チームと稼働そのものに価値を置く
- 柔軟な仕様変更: 理解が進むにつれて、仕様や優先順位を柔軟に変更できる
- 知識の蓄積: AIが蓄積した知識や文脈が、チームに継承され続ける
- 長期的な関係: 長期的なパートナーシップを前提とし、継続的な改善が可能

従来の請負型契約では、プロジェクトごとに知識がリセットされ、同じような問題を繰り返し解決する必要がありました。しかし、ラボ型契約では、知識が蓄積され続け、過去の経験が次のプロジェクトに活かされます。これにより、開発の質と効率が、時間とともに向上していきます。契約形態の選択は、単なる法的な問題ではありません。開発の進め方、知識の蓄積、チームの成長、長期的な価値創

造に、大きな影響を与えます。AI時代の開発においては、ラボ型契約が、持続可能な開発体制を構築する上で、不可欠な要素となります。

第8章｜それでも失敗する企業の共通点

ここまでの章で、オフショア活用を成功に導くためには、日本側の開発構造の改革や契約形態の見直しといった、開発作業そのもの以外の要素が重要であることを述べてきました。

しかし、これらを理解していても、なお失敗する可能性はあります。

構造を理解せず、単価だけで判断する。日本側の準備を省き、丸投げを期待する。AIを万能な魔法と誤解する。こうした姿勢では、分業モデルは機能しません。

8.1 単価を最優先とする企業

多くの企業が、オフショア開発を導入する際に単価だけで判断します。確かに、コストは重要な要素ですが、単価だけで判断すると、長期的な価値を見失います。

単価が安いからといって、品質が低かったり、コミュニケーションコストが高かったりすれば、結果的にコストが増大します。また、単価が安いからといって、知識が蓄積されない体制では、長期的な価値が生まれません。

AIとラボ型契約を組み合わせることで、知識が蓄積され長期的な価値が生まれます。この価値を理解せず、単価だけで判断する企業は、分業モデルの真の価値を得ることができないため、総合的な価値を重視することが重要です。

8.2 丸投げを望む企業

日本側の準備を省き、丸投げを期待する企業も、失敗しがちです。AIが情報を構造化し、理解を支援するとはいえ、日本側が何も準備せず、すべてをオフショアに任せることはできません。分業モデルを成功させるためには、日本側が以下のような準備を行う必要があります。

- 役割の明確化: 日本側とオフショア側の役割を明確に定義する
- 情報の構造化: AIが理解できる形で、情報を構造化する
- 判断基準の明文化: 判断基準や品質基準を明文化する
- 継続的なコミュニケーション: 定期的なコミュニケーションにより、理解を共有する

これらの準備を行わず、丸投げを期待する企業は、オフショアチームが適切に作業を進めることができず、結果として失敗を招くことになります。

8.3 AIを万能と期待する企業

AIを万能な魔法と誤解する企業も、失敗しがちです。AIは強力なツールですが、すべてを自動的に解決するわけではありません。AIは、情報を構造化し、理解を支援し、判断を促進するツールです。最終的な判断は、依然として人間が行う必要があります。

AIを導入すれば、すべてが自動的に解決されると期待する企業は、現実とのギャップに失望し、AIの導入を諦めてしまいます。AIの限界を理解し適切に活用することが重要です。

8.4 成功している企業の共通点

成功している企業は、小さく始め、状況を可視化し、改善を前提にワンチームで進めています。AIは企業の姿勢をそのまま映し出す鏡です。

- 小さく始める: いきなり大規模な導入を行うのではなく、小さなプロジェクトから始める
- 状況を可視化する: AIを活用して、開発の状況を可視化し、問題を早期に発見する
- 改善を前提とする: 完璧を目指すのではなく、継続的な改善を前提とする
- ワンチームで進める: 日本側とオフショア側が、対立するのではなく、協力して進める

分業モデルの成功は、技術やツールだけでなく、企業の姿勢によって決まります。構造を理解し、適切に準備し、継続的に改善する姿勢を持つ企業は、分業モデルを成功させることができます。

第9章｜2026年型・三者分業モデルの全体像

ここまで述べてきた要素を統合すると、2026年型の開発体制は、日本・ベトナム・AIによる三者分業モデルに行き着きます。このモデルは、単なる役割分担ではなく、それぞれが最も価値を発揮できる領域に集中するための構造設計です。

日本側は、経営判断、事業設計、プロダクトオーナーシップ、仮説検証といった意思決定と創造に責任を持ち、ベトナム側は、実装、解析、改善、既存保守といった構築と安定化を担います。AIは、この二者をつなぐ存在として、翻訳、要約、構造化、影響分析、検証といった横断的な役割を担い、人と人の間に生じていた摩擦を吸収します。AIによって構造化された情報を前提にすることで、国や個人差に依存しない再現性のある開発が可能になります。

この三者分業モデルによって、初めてオフショア開発体制は持続的にスケールします。誰か一人に負荷が集中することなく、理解が蓄積され、次の意思決定につながる循環が生まれます。

再配置を実現するためには、以下のような取り組みが必要です。

- 役割の明確化: 日本側、ベトナム側、AIの役割を明確に定義する
- 情報の構造化: AIにより、情報を構造化し、理解の土台を揃える
- 契約形態の選択: ラボ型契約により、知識の蓄積を促進する
- 継続的な改善: 小さく始め、状況を可視化し、継続的に改善する

これらの取り組みにより、再配置が実現し、持続可能な開発体制が構築されます。AIとオフショアは、人件費削減の手段ではなく、日本企業が2026年以降も開発力を維持するための現実的な選択肢です。限られた人材を効率的に活用し、創造性の高い仕事に集中させることで、日本企業は、グローバルな競争において、優位性を維持することができます。

おわりに

本ホワイトペーパーでは、2026年を見据えた日本企業のIT開発における構造的な課題と、その解決策としてのAI駆動開発とベトナムオフショアを組み合わせた三者分業モデルについて、9章にわたって詳しく解説してきました。

重要なのは、AIやオフショアを単独の施策として捉えるのではなく、開発の構造そのものを再設計する視点です。情報の構造化、役割の明確化、知識の蓄積、継続的な学習。これらの要素を統合することで、初めて持続可能な開発体制が構築されます。

本モデルは、限られた人材を最も価値の高い業務に集中させるための再配置戦略です。ベトナム側とAIは、構築と安定化を担い、理解を加速させるインフラとして機能することで、日本側のエンジニアは、創造性の高い仕事に集中でき、技術者の価値そのものを引き上げることができます。

2026年以降、日本企業が開発力を維持し、グローバルな競争において優位性を保つためには、このような構造的な再設計が不可欠です。本ホワイトペーパーが、その一助となれば幸いです。

当社事業「スマラボ」のご紹介

― AI時代の現実解としてのベトナムオフショア ―

スマラボは、20年以上にわたるオフショア開発の経験を基盤に、日本人による上流工程やオフショア活用支援、ベトナム現地に常設したラボ体制、そしてAI活用を前提とした開発プロセスを組み合わせることで、「理屈ではなく、実際に回るAI×オフショア」を提供しています。

スマラボが重視しているのは、個別技術や一時的な人員確保ではありません。要件・仕様・判断プロセスを構造化し、知識がチームとAIの双方に蓄積され続ける状態を前提とした、持続可能な開発体制です。

提供サービス

- **プロジェクト立ち上げ支援**（体制設計・役割定義・初期整理）
- **ラボ型開発による継続的な新規開発・改善**
- **既存システムの保守・解析・構造化支援**
- **日本人プロジェクトコーディネータ × ベトナムエンジニア × AIによる日越ハイブリッド体制**

AIを前提にした分業設計とラボ型体制を組み合わせることで、日本側は判断と創造に集中し、オフショアは実装と改善を担い、AIがその橋渡しをする。スマラボは、この三者分業モデルを実行レベルで支えるためのサービスです。

HP：https://sma-labo.jp/

発行者

＜会社概要＞

株式会社アイディーエス

https://www.ids.co.jp/

独立系のシステムインテグレーターとして、情報系システム構築を専門に、AWSをはじめとした各種システムの設計・構築から運用、活用支援まで、数多くの提案・開発実績を有しています。

本社所在地	〒105-0014 東京都港区芝2-3-18 YM芝公園ビル 5階
代表取締役	中野 貴志
事業内容	ベトナムオフショアラボ開発、システム開発、AWS請求代行、人材派遣

＜お問い合わせ先＞

株式会社アイディーエス　スマラボ事業部

TEL：03-5484-7811

E-mail：smalabo@ids.co.jp