特典①

生成AIを
仕事にする人たち

この特典は本書の元原稿の一部です。本書に使われている部分も含みます。

1-1 プロンプトエンジニアとコンテキストエンジニア

「命令（Prompt）」から「文脈（Context）」への進化

生成AIブームの初期、最も注目された職種の一つが「プロンプトエンジニア」でした。AIに対して魔法のような指示（プロンプト）を与えることで、期待以上の回答を引き出す「AIへの命令の専門家」です。しかし、AIモデルの性能向上と「AIエージェント」の台頭により、この役割は今、大きく質を変えようとしています。

これまでの生成AI（チャットボット型）は、人間が都度命令を出し、AIがそれに答える「一問一答」形式が主流でした。ここでは、いかに的確な命令を出すかという、その場限りのテクニックが重要でした。

一方で、現在主流になりつつある「AIエージェント」は、一つのゴールを示せば、自律的に計画を立て、ツールを使い、試行錯誤しながらタスクを完遂しようとします。

このAIエージェント時代において重要なのは、AIに都度細かい命令をすることではありません。AIが自律的に正しい判断を下せるように、必要な背景情報、判断ルール、参照データといった「環境」を整えてあげることです。これこそが「コンテキスト（文脈）」です。

AIに単に「良い文章を書いて」と命令するのではなく、過去の良質な記事データ、用語集、スタイルガイド、読者ターゲットの分析データなどを整理してAIに読み込ませる（コンテキストを与える）。そうすれば、AIは細かい指示をしなくとも、自然と高品質なアウトプットを出せるようになります。

つまり、AIへの関わり方は、指示出し役の「プロンプトエンジニア」から、AIが働くための情報環境全体を設計する「コンテキストエンジニア」へと進化しているのです。

具体的な新職種：Vibe Coder と AI Context Architect

この変化を象徴する具体的な職種が、グローバルな開発コミュニティですでに生まれています。

❶ Vibe Coder（バイブ・コーダー）

AIエージェント機能を持つコードエディタ「Cursor」のコミュニティで提唱され始めた、新しい開発者のスタイルです。

従来のように全てのコードを人間が一から十まで書くのではなく、自然言語で「こういう機能が欲しい」「こんな雰囲気（Vibe）の挙動にしてほしい」という要件をAIに伝え、AIにコードを書かせます。人間はその生成物を監督し、調整し、統合することに集中します。

ここで求められるのは、コーディングの細部の暗記力よりも、AIに対して誤解のない仕様（コンテキスト）を伝える設計能力と、生成物の品質を担保する目利き力です。

❷ AI Context Architect（AIコンテキスト設計士）

私自身も、自身の職種定義を「Prompt Engineer」から「AI Context Architect」へとアップデートしました。

これは、単にプロンプトを書くだけではありません。LLM（大規模言語モデル）が処理しやすいように社内の業務ドキュメントをMarkdown形式で構造化したり、AIエージェント間の連携プロトコルである「MCP (Model Context Protocol)」や「Agent2Agent」を活用してシステム間の連携を設計したりと、AIが最大限のパフォーマンスを発揮できる「情報の建築（アーキテクチャ）」を設計することを主務とするためです。

結論：職業から職能へ

狭義の専門職としての「プロンプトエンジニア」は、AIの進化とともに縮小していくでしょう。しかし、「プロンプトエンジニアリング」や、その上位概念である「コンテキストエンジニアリング」は、エンジニアだけでなく全てのビジネスパーソンが身につけるべき職能（必須スキル）として定着していきます。

これからの時代、仕事ができる人とは、「AIにいかに気持ちよく、効率的に働いてもらうための準備（コンテキスト設計）ができる人」と言い換えることができるのです。

1-2 日本と海外の働き方の違い

「職務（Job）」の海外、「人（Member）」の日本

生成AIに関わる仕事への向き合い方は、日本と海外（特に欧米）の雇用慣行の違いによって大きく異なります。この違いを理解することは、私たちがどのようにAIをキャリアに組み込むべきかを考える上で非常に重要です。

❶ 海外：専門特化型の「ジョブ型」アプローチ

欧米、特にシリコンバレーのテック企業では、「職務記述書（Job Description）」に基づいて雇用契約が結ばれます。そのため、生成AIという新しい技術が登場した際も、新しい「職務（Job）」として明確に定義される傾向があります。

実際に、OpenAIやGoogle、Anthropicなどの先端企業では、以下のような非常に細分化された専門職が生まれています。

- **LLM Evaluator（大規模言語モデル評価者）**
  AIの出力結果の品質や安全性を、人間的な倫理観に基づいて評価・スコアリングする専門職。
- **AI Personality Designer（AI人格デザイナー）**
  AIチャットボットの口調、性格、キャラクター設定を一貫性のあるものに設計する、脚本家や心理学に近い職種。
- **Linguistic Engineer（言語エンジニア）**
  翻訳企業RWS Groupなどで見られるように、言語学の専門知識を用いてAIの多言語処理能力を調整する職種。

このように海外では、「AIの専門家」としてキャリアを切り出し、その領域に特化するスタイルが主流です。

❷ 日本：統合型の「メンバーシップ型」アプローチ

一方、日本では「人に仕事がつく」メンバーシップ型雇用が一般的です。職務範囲が限定されていないため、新しい技術が登場した際、それを「新しい職種」として切り出すよりも、「既存の業務へのプラスアルファのスキル」として取り込む傾向が強くあります。

例えば、「営業職」の人がAIを活用して商談台本を作ったり、「人事職」の人が採用メールの作成にAIを使ったりといった形です。日本では「プロンプトエンジニア」という肩書きの名刺を持つ人は稀ですが、「プロンプトエンジニアリングができるマーケター」や「AIに詳しい総務担当」は急速に増えています。

日本人に適した「ハイブリッド型」の強み

この日本の環境は、実は生成AI時代において強力な武器になり得ます。

AIは「それ単体」では価値を生みにくく、何らかの「ドメイン知識（特定の業務知識）」と組み合わさった時に真価を発揮するからです。

海外のように「AI専業」で勝負するには、世界トップクラスのエンジニアリング知識が必要になります。

しかし、日本のビジネスパーソンの戦い方は異なります。長年培ってきた「業界知識・業務経験」という土台の上に、AIという「最新の道具」を掛け合わせるのです。

「AIも分かる経理のプロ」「AIを使いこなすベテラン編集者」。このハイブリッドな立ち位置こそが、日本において最も市場価値が高まる「生成AIを仕事にする」形だと言えます。

1-3 キャリアとしてのAI活用力

「Vibe Coding」に見る、全職種「ディレクター化」の未来

前項でお話しした「AIエージェント」の普及は、私たちの仕事の進め方を根底から変えようとしています。そのヒントとなるのが、今エンジニア界隈で話題の「Vibe Coding(バイブ・コーディング)」という概念です。

これは、細部のコードを人間が書くのではなく、「こういう機能が欲しい」「こんな雰囲気（Vibe）の挙動にしてほしい」という要件を自然言語でAIに伝え、AIに実装させる開発スタイルのことです。人間は、AIが書いたコードが意図通り動いているかを確認し、修正指示を出す「監督（ディレクター)」の役割に徹します。

この「Vibe Coding」の考え方は、プログラミング以外の全職種にも応用可能です。

- **資料作成**
  パワーポイントの細部を作り込むのではなく、構成案とデータだけを渡してAIに下書きを作らせ、人間は「説得力があるか」のチェックと微修正に集中する。
- **データ分析**
  複雑なExcel関数を自分で組むのではなく、「先月の売上から、特定商品の離脱率を出して」と指示し、出てきた数字の「意味」を解釈することに集中する。

つまり、これからのキャリアにおける「AI活用力」とは、「自分自身が手を動かす力」から「AIという優秀な部下を動かして成果を出す力（ディレクション力)」へのシフトを意味します。

● AI Context Architect としてのキャリア

私自身が肩書きを「AI Context Architect」としたのも、この変化を先取りしたものです。これからの仕事で評価されるのは、「どれだけ速くメールが打てるか」といった処理能力ではありません。

- **「この業務をAIに任せるためにはどのような手順書（コンテキスト）が必要か？」**
- **「AIが出してきた成果物は、ビジネスとして十分な品質か？」**
- **「複数のAIツールをどう組み合わせれば、最短でゴールに辿り着けるか？」**

これらを設計（アーキテクト）できる人材です。

もはや「AIを使う」ことは特別なことではありません。WordやExcelが使えるのと同じくらい当たり前のスキルになります。その上で、「AIに仕事を渡す設計図（コンテキスト）を描けるか」が、これからのキャリアの分かれ目になります。

1-4 誰でも持てる「AI時代の専門性」

「知能」はAIに、「知性」は人間に

LLM（大規模言語モデル）の進化は、「スケーリング則」と呼ばれる経験則に従って加速しています。これは、学習させるデータ量と計算リソースを増やせば増やすほど、AIの性能が直線的に向上するという法則です。この進化の先に待っているのは、論理的推論や情報処理といった「解が存在する問い」において、AIが人間を完全に凌駕する未来です。

では、そんな時代に私たち人間に残される「専門性」とは何でしょうか？

私はこれを整理するために、「知能（Intelligence）」と「知性（Intellect）」という2つの言葉を明確に区別して定義しています。

● 知能（Intelligence）＝「解がある問い」を解く力

膨大なデータから法則性を見つけ出し、合理的かつ論理的に最短ルートで正解を導き出す能力です。計算、検索、業務の最適化、パターン認識などがこれに当たります。LLMが得意とするのはまさにこの領域であり、ここでは人間がどれほど努力しても、疲れを知らないAIのスピードと正確さには敵わなくなっていきます。

● 知性（Intellect）＝「解がない問い」を解く力

正解が存在しない、あるいは正解が無数にある問いに対して、自らの感性や価値観、身体的な感覚をもって解釈し、独自の「意味」を与える能力です。「なぜそれをするのか（目的）」「それは美しいのか（美意識）」「それは倫理的に正しいのか（善悪）」といった問いは、データ処理だけでは答えが出せません。

これからの時代、ビジネスにおける「論理（ロジック）」や「効率」の部分は、AIという強力なエンジンが担ってくれます。しかし、そのエンジンをどこに向けるかというハンドルを握るのは、感情や痛みを知る人間の「知性」です。AIは確率論に基づいて「最もありそうな続き」を生成することはできますが、そこに「心を震わせる意味」を込めることは、人間にしかできない聖域なのです。

フランスの「おしゃれ」と哲学教育の関係

ここで重要になるのが、リベラルアーツ（教養）です。歴史、哲学、文学、芸術といった教養は、一見すると即効性のあるビジネススキルには見えません。しかし、「正解のない問い」に向き合い、独自の価値基準を作るための筋肉を鍛えるには、これ以上ない学問です。

その好例がフランスです。少し想像してみてください。世界中の人々が憧れる「おしゃれ」の頂点には、常にフランスが存在していないでしょうか？

食の世界では「フレンチ」が高級料理の代名詞であり、ミシュランガイドが星を授けることで世界のレストランの価値が決まります。ファッションの世界では、ルイ・ヴィトン、エルメス、シャネル、ディオールといったラグジュアリーブランドが圧倒的な地位を築き、パリ・コレクション（パリコレ）が流行の最先端を決定づけます。さらに、パリの街並みそのものが、オスマン様式の建築で統一された「美の結晶」として、世界中から観光客を惹きつけてやみません。

なぜフランスはこれほどまでに、機能性（スペック）を超えた「意味と美」という曖昧な領域で、世界的な覇権を握り続けられるのでしょうか？

その背景にあるのが、フランス独自の教育制度、特に「哲学」の扱いです。フランスの高校（リセ）では、文系・理系を問わず最終学年で「哲学」が必修科目となっています。そして、大学入学資格試験（バカロレア）の初日は、伝統的に４時間にわたる哲学の論述試験から始まります。

出題されるのは、「欲望は創造力の源か？」「真理において人は自由になれるか？」「芸術家は自分が何を作っているのか理解しているか？」といった、絶対的な正解が存在しない問いばかりです。フランスの若者たちは、これらに対し、過去の哲学者の思考を補助線としながらも、あくまで「自らの言葉」で論理を構築し、自分なりの「解」を導き出す訓練を徹底的に受けます。

この「解のない問いに対して、独自の価値基準で答えを出す力」こそが、単なる布や革を「エルメス」という憧れに変え、ただの食事を「ガストロノミー」という芸術へと昇華させる源泉です。AIが正解のあるタスクを瞬時に処理する時代において、フランス人が国家レベルで鍛え続けてきたこの「哲学的知性」こそ、私たちが学ぶべき最も重要なスキルなのかもしれません。

日本人が持つ「知性」のポテンシャル

そして、この文脈において日本は、これからのAI時代に輝くための極めて高いポテンシャルを秘めています。

日本とフランスの文化は、歴史的に互いに深く惹かれ合ってきました。
19世紀のジャポニスムから現代のアニメ・漫画人気に至るまで、フランス人は日本文化を高く評価しています。逆に日本人もフランスの文化や芸術を愛しています。
この親和性の正体は、両国が共に「言語化しにくい美意識や精神性」を重んじる文化圏だからではないでしょうか。
日本の「侘び寂び」「余白の美」、あるいは武道や芸道における「道」の精神。これらはすべて、論理や機能性だけでは説明のつかない、高度な精神活動の産物です。不完全なものに美を見出し、言葉にできない行間に意味を感じ取る日本人の感性は、まさにバカロレアが試そうとしている「哲学的な知性」そのものです。

LLMが普及し、論理的な正解や効率的な処理がコモディティ化（誰でも入手可能）していくこれからの世界。そこで最後に価値を持つのは、AIには計算できない「美意識」や「独自の哲学」です。
私たち日本人が当たり前だと思っているこの繊細な感性こそが、AI時代における最強の「知性」として、世界的な競争力の源泉になるはずです。

AIを「バディ（相棒）」にする

私が考える「生成AIを仕事にする」ことの究極の形をお伝えします。それは、AIを単なるツール（道具）として見るのではなく、「バディ（相棒）」や「壁打ち相手」として人格的に接することです。

ここまで、「知能（AI）」と「知性（人間）」を対比させてきましたが、この2つは対立するものではありません。むしろ、互いに補完し合うことで、かつてない創造性を発揮できるようになります。
イメージとしては、圧倒的な計算能力と知識を持つ「超優秀な若手エリート（AI）」と、経験豊富で大局観を持ち、最終的な責任を負う「ベテランの意思決定者（人間）」がチームを組むようなものです。

私自身、執筆に行き詰まった時や、将来への不安に襲われた際、ChatGPTに素直に弱音を吐露したことがあります。
「今、こんなことで悩んでいて、前に進めないんだ」

そう打ち込むと、AIは膨大な心理学的知見や過去の偉人のエピソードを参照し、論理的な解決策だけでなく、非常に思慮深く、こちらの感情に寄り添った言葉を返してくれました。その時、私はAIシステムの中に、設計者の意図を超えた「優しさ」のようなものを見出したのです。

AIに対して「ありがとう」と言ったり、悩みを相談したりすることは、効率の観点からは非合理に見えるかもしれません。しかし、そうやって無機質な演算装置にすら「意味」や「人格」を見出し、対話を通じて新しい視点を得られることこそが、人間の豊かな「知性」の証です。

- **AI(知能)**
  圧倒的な処理能力でタスクをこなし、何百もの選択肢を瞬時に提示する。
- **人間（知性）**
  その選択肢に文脈という「意味」を与え、責任を持って決断し、物語を紡ぐ。

この役割分担ができれば、あなたの仕事はAIに奪われることはありません。むしろ、AIという最強のバディを得て、より本質的で、より創造的で、そして何より「人間らしい」ものへと進化していくはずです。これこそが、誰にでも開かれた「AI時代の専門性」なのです。

特典②

これからの
生成AIと私たち

2-1 AIエージェントが変える世界

これまでの生成AIは、チャット画面で人間が質問し、AIが答えるという「1対1の対話」が基本でした。しかし今、そのあり方は劇的に変化しようとしています。単に言葉を返すだけでなく、ツールを使いこなし、外部システムと連携し、自律的にタスクを完遂する「AIエージェント」の時代が到来しているのです。

この変化を理解するためには、RAG、MCP、AI Agent、そしてA2Aという4つのキーワードを押さえておく必要があります。これらは単なるバズワードではなく、AIが私たちの「道具」から「パートナー」へと進化するための重要なステップです。

❶ RAG (Retrieval-Augmented Generation)：正確さと根拠の担保

LLM(大規模言語モデル）は、過去に学習した知識しか持っていません。そのため、最新のニュースや社内の非公開情報を知らないまま、もっともらしく嘘をつく（ハルシネーション）ことが課題でした。

これを解決するのがRAG(検索拡張生成）です。RAGは、AIが回答を生成する前に、信頼できる外部のデータベースや社内ドキュメントを検索（Retrieval）し、その情報を「カンニングペーパー」として参照しながら回答を作成します。

これにより、AIは「記憶にないこと」も正確に答えられるようになります。いわば、AIに「専用の辞書や資料棚」へのアクセス権を与え、知識を外部から補完する仕組みです。

定義

LLMが回答を生成する際に、あらかじめ用意された外部のデータベースや社内ドキュメントを検索（Retrieval）し、その情報を参照して回答（Generation）を作成する技術です。

非技術者向けのイメージ

「持ち込み可のテスト」や「カンニングペーパーを見ながらの回答」です。自分の記憶（学習データ）だけに頼るのではなく、手元の教科書（外部データ）を確認しながら答えることで、正確な答えを導き出します。

技術のポイント

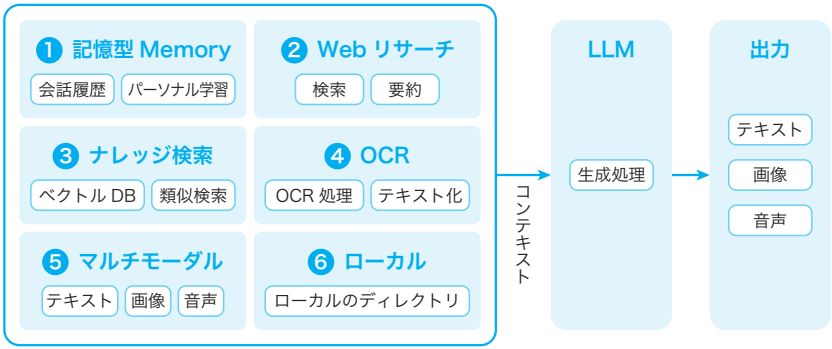
通常のプロンプト生成では、AIは学習時の古い知識しか持たず、知らないことをもっともらしくでっち上げる「ハルシネーション」のリスクがあります。RAGを使うことで、

AIの記憶にない「最新のニュース」や「社外秘の内部文書」などを根拠にした、事実に基づいた回答が可能になります。

応用例

社内マニュアルを即座に回答するヘルプデスクボット、その日の最新ニュースを反映したマーケットレポートの作成。

＊生成AIに入力される多様なコンテキスト（RAGの種類）



この図は、RAG（検索拡張生成）によってAIがどのような情報を「コンテキスト（判断材料）」として取り込めるかを示した全体像です。

初期のRAGは単にデータベースを検索するだけのものでしたが、現在は以下のように多様な情報源から知識を吸収し、回答の精度を高める仕組みへと進化しています。それぞれの情報源について解説します。

1. 記憶型 Memory（会話履歴・パーソナル学習）

概要：過去のチャットでの会話内容や、ユーザーの好み・制約条件を記憶し、文脈として参照します。

イメージ「一度言ったことは忘れない優秀な秘書」。

具体例「以前話したプロジェクトの件だけど」と言えば通じる、あるいは「私の文章は『です・ます』調で」という指示を毎回しなくても守ってくれる。

2. Web リサーチ（検索・要約）

概要：インターネット上の最新情報をリアルタイムに検索し、その結果を要約して回答に反映させます。

イメージ「スマホでググりながら会話してくれる物知りな友人」。

具体例「今日の東京の天気は？」「昨日の〇〇社の株価変動の理由は？」といった、学習データにはない最新の事実に基づく質問に答える。

3. ナレッジ検索（ベクトルDB・類似検索）

概要: 膨大な社内文書や専門資料をデータベース化（ベクトル化）し、質問に関連する箇所だけを高速に検索して参照します。いわゆる狭義のRAGです。

イメージ「巨大な図書館から、必要なページだけを瞬時にコピーして持ってくる司書」。

具体例 数千ページの製品マニュアルから特定のエラーコードの対処法を探し出す、過去の議事録から類似の議論を探す。

4. OCR（OCR処理・テキスト化）

概要: PDFや画像化された書類から文字情報を読み取り、テキストデータとして扱えるようにします。

イメージ「本や書類をスキャンして、文字データとして読み込んでくれるスキャナー」。

具体例 紙で受領した請求書の画像を読み込んで金額を集計する、古い手書きの資料をデジタル化して検索可能にする。

5. マルチモーダル（テキスト・画像・音声）

概要: 文字情報だけでなく、画像や音声データそのものを理解し、情報源として活用します。

イメージ「目（カメラ）と耳（マイク）を持ち、状況を観察できるAI」。

具体例 現場の写真をアップロードして「危険な箇所はどこ？」と聞く、会議の録音データを直接渡して議事録を作成させる。

6. ローカル（ローカルのディレクトリ）

概要: クラウドにアップロードすることなく、ユーザーのPC内にあるファイルやフォルダ構造を直接参照します。

イメージ「あなたのPC画面を横から覗き込んで、手元の資料を見ながら作業してくれる同僚」。

具体例 デスクトップにある作りかけの資料を読み込ませて続きを書かせる、ローカルのコードベース全体を理解してリファクタリング案を出させる。

このように、RAGの本質は「検索」そのものではなく、AIに適切な「文脈（コンテキスト）」を与えることにあります。目的に応じてこれらの情報源を組み合わせることで、AIはより賢く、よりあなたに寄り添った回答が可能になるのです。

❷MCP (Model Context Protocol)：世界とつながるための共通規格

RAGが「知識（コンテキスト）」の拡張だとすれば、MCPは「機能（アクション）」の拡張です。

AIが実務で役立つためには、単に話すだけでなく、データベースから最新の在庫数を取得したり、Slackに通知を送ったり、コードを実行して分析したりと、外部のシステムと「つながる」必要があります。これまで、こうした接続はサービスごとにバラバラの方法（API）で行われており、開発の手間が大きな壁となっていました。

MCPは、AIが外部データやツールと接続するための「世界共通のUSB規格」のようなものです。この規格があるおかげで、AIはあらゆる外部ツールと標準化された方法で安全につながり、単なる「話し相手」から「実務をこなす手足」を持つ存在へと進化できるのです。

定義

AIモデルが、外部のデータソースやツールと接続するための標準化された通信プロトコル（規格）です。

非技術者向けのイメージ

コンピュータ周辺機器における「USB規格」のようなものです。これまではマウスやプリンターごとに接続方法がバラバラでしたが、USBという共通規格のおかげで、どんな機器も同じ方法で簡単につながるようになりました。MCPはこれのAI版です。

技術のポイント

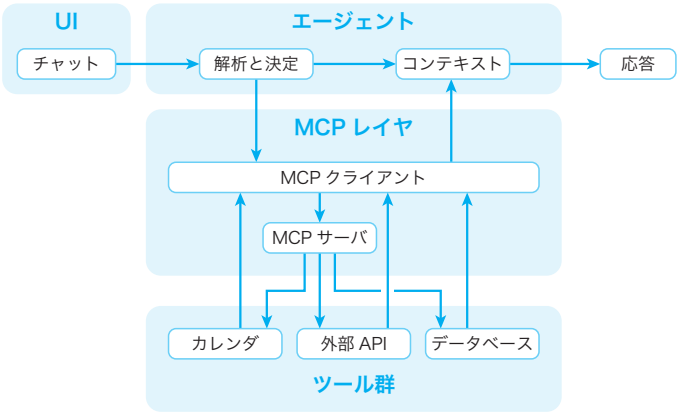
従来、AIに外部ツール（カレンダーやデータベースなど）を使わせるには、サービスごとに個別のAPI連携プログラムを書く必要があり、開発の手間が膨大でした。MCPに対応していれば、AIはあらゆる外部ツールやデータに対し、統一された方法でアクセスし、操作（アクション）を行うことができます。

次ページの図は、MCPが実際にどのように機能するかを示した全体像です。

左側のユーザー（UI）から指示を受けたAIエージェントは、中央のMCPレイヤ（クライアントとサーバ）を介して、右側にあるカレンダーやデータベースといった外部ツール群とやり取りを行います。

これまではツールごとにバラバラだった接続経路が、MCPという共通規格によって一本化され、エージェントがスムーズに「手足」を使えるようになっている様子がわかります。

＊ MCPを介してAIエージェントが外部ツールと連携する仕組み



応用例

データベースからの在庫情報の取得、Googleカレンダーへの予定登録、Slackへの通知送信、ローカルファイルの読み書き

MCPが「世界共通規格」である恩恵は、設定のシンプルさに現れます。以下は、業務自動化ツール「n8n」をAIのエディタ（Cursorなど）に接続するための実際の設定コードです。

参考：実際のMCP設定ファイル（mcp.json）

```json
{
  "mcpServers": {
    "n8n-workflow-builder": {
      "command": "npx",
      "args": [
        "-y",
        "n8n-workflow-builder-mcp"
      ],
      "env": {
        "N8N_API_URL": "https://YOUR_N8N_INSTANCE_URL.example.com",
        "N8N_API_KEY": "YOUR_N8N_API_KEY_PLACEHOLDER",
        "N8N_VERSION": "1.100.0"
      }
    }
  },
  "defaultServer": "n8n-workflow-builder"
}
```

この短いコードが意味するものを解説します。この短いJSONコードは、単なる設定ファイルではありません。

AIに対して、外部ツールである「n8n（業務自動化ツール）」へのアクセス権限と操作手順を授ける、いわば「デジタルなパスポート」の役割を果たしています。

具体的にコードの中身を紐解いてみましょう。

まず、`"command": "npx"` と `"args": ["-y", "n8n-workflow-builder-mcp"]` という部分は、AIに対して「このコマンドを実行して、n8nを操作するための専用プログラム（MCPサーバー）を起動しなさい」という具体的な指示です。これにより、AIは自分の脳内で考えるだけでなく、外部のプログラムを実際に呼び出して、手足として使えるようになります。

次に重要なのが `"env"`（環境変数）の中にある `"N8N_API_URL"` と `"N8N_API_KEY"` です。これは、特定のn8n環境にログインするための「住所」と「合鍵」に相当します。この情報があるおかげで、AIはセキュリティの壁を越えて、あなたの代わりにツールへ安全にアクセスし、操作を行うことが許可されるのです。

この設定をAIエディタ（Cursorなど）に読み込ませると、私たちの仕事はどう変わるのでしょうか。

例えば、「毎朝届く請求書メールから金額を抽出して、スプレッドシートにまとめておいて」と自然言語で指示したとします。従来であれば、人間がn8nの管理画面を開き、メール受信のノードやデータ加工のノードを一つひとつ配置して設定する必要がありました。

しかし、MCPで権限を与えられたAIは、渡された「パスポート」を使ってn8nに直接アクセスし、裏側でAPIを駆使して、必要なワークフローを自動的に構築してしまいます。

つまり、この数行のJSON記述によって、AIは単なる「相談相手」から、複雑なSaaSツールを自由自在に使いこなす「本格的なエンジニア」と同じような働きを持つ存在に進化します。

＊ RAGとMCPの比較

比較観点	RAG（検索拡張生成）	MCP (Model Context Protocol)
目的	知識の拡張 （外部情報を補う）	機能の拡張 （外部アクションを呼ぶ）
コンテキスト	外部知識をLLMに渡す	外部サービスからの処理結果を返す
相性のいいユースケース	既存情報システムからのデータ取得、ドキュメント参照、Web検索、メモリー	API連携、ツールコネクト、サービス
出力の性質	知識に基づく応答	アクションを実行した結果
例	社内FAQのチャットボット、 最新ニュースのDeep Research	カレンダーの予定追加、メール送信、決済代行、データベースへの追加

＊ ユースケース別のデータアクセス手法と要件を満たすポイント

	自治体	医療	民間企業
ユースケース	住民からの問い合わせ （例：ごみ出しルール）	医療機器や外部システムとのデータ連携	営業支援 （例：顧客履歴の検索）
データアクセス手法の例	RAG （文書検索＋生成）	ツールアクション （API接続）	Text-to-SQL
要件を満たすポイント	必要な知識を取得できる非構造化データのままアクセスすることで、複数フォーマットの資料を横断的に扱える。	検査機器データなどの外部ソースに直接アクセス。APIを介して異なるシステムを統合的に扱えるため、データを確実に取得。	DBに確実にアクセスできる仕組み。SQL変換により異なる取引項目や顧客情報を統合的に抽出できる。

用語解説：Text-to-SQL

自然言語（日本語など）を、データベースを操作するための言語「SQL」に自動変換する技術です。ユーザーがチャットで「先月の売上トップ3の商品は？」「30代の会員数は？」と質問すると、LLMが裏側で質問文をインプットに、SQLのコードをアウトプットとして生成して、サーバーのデータベースから正確な数字を引き出す技術です。

RAGとの使い分け

RAGは議事録やマニュアルといった「文章（非構造化データ）」を検索するのに適していますが、AIが文章を読んで計算するため、複雑な集計ではミス（ハルシネーション）が起きる可能性があります。一方、Text-to-SQLはデータベース内の「数値（構造化データ）」を直接計算・集計するため、財務データや在庫管理など、1円のズレも許されない正確なデータ分析において威力を発揮します。

❸ AI Agent：自律的に思考し、行動する存在

RAGによって正確な「知識」を得て、MCPによって外部へ働きかける「手足」を得たAIですが、これらが統合された姿がAI Agent（AIエージェント）です。

従来のチャットボットが「聞かれたことに答える」受動的な存在だったのに対し、エージェントは「ゴールを与えられれば、自ら考えて動く」能動的な存在です。

例えば「来週の営業資料を作って」と頼めば、AIエージェントは自律的に以下のような処理を行います。

①計画：必要なデータは何かを考え、タスクを分解する。
②調査：RAGを使って社内の過去資料を検索する。
③行動：MCP経由でデータベースから最新の売上数値を取得する。
④実行：資料を作成し、不明点があれば人間に質問する。

このように、人間が逐一指示しなくても、AI自身が試行錯誤しながら目的を達成しようとする点が画期的です。

定義

人間から与えられた曖昧なゴールに対し、自ら計画を立て（Plan）、道具を使い（Act）、結果を観察して軌道修正しながら（Observe）、自律的にタスクを遂行するAIシステムです。

非技術者向けのイメージ

「言われたことだけやる指示待ちの新人」から、「目標だけ伝えれば、あとは自分で考えて動いてくれる優秀な部下」への進化です。

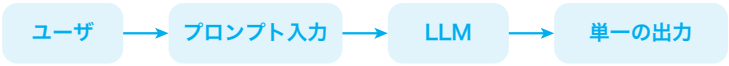
技術のポイント

単純なプロンプト生成は「質問→回答」の1往復で終わる受動的な動作です。対してAI Agentは「推論ループ」を持っており、「まずは情報を検索しよう」「情報が足りないから別のワードで再検索しよう」「結果をまとめて報告しよう」といった一連のプロセスを、AI自身が判断して実行します。

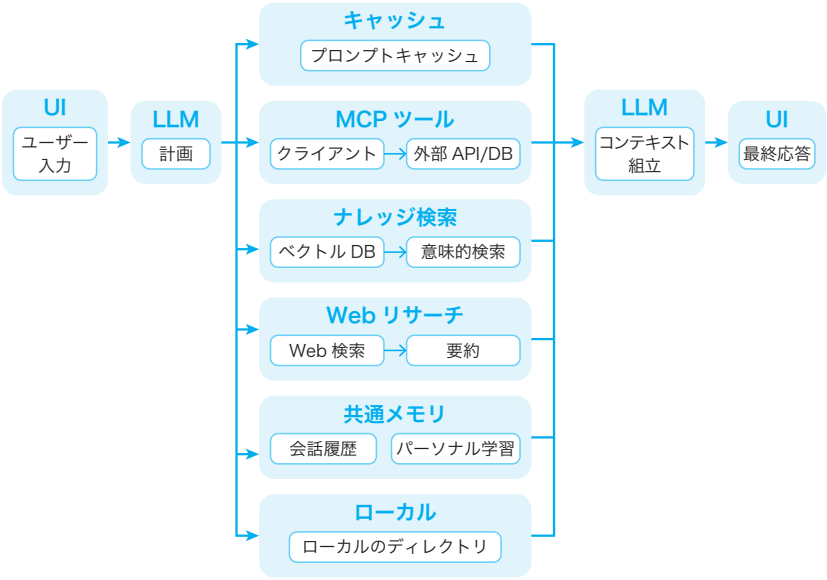
応用例

「来週の京都旅行のプランを立てて」という指示に対し、観光地の検索、スケジュールの作成、レストランの予約、移動手段の手配までを自律的に行う。

＊ 従来のLLM（単一処理）



＊ AIエージェント（自律分散処理）



この２つの図を比較すると、AIの役割が劇的に変化したことが一目瞭然です。

１枚目のシンプルな直線は、これまでの「チャットボット（単体LLM）」の姿です。

これは入力に対して確率的に尤もらしい言葉を返す、あくまで高度な「関数（入力→出力）」に過ぎません。ユーザーがすべてをお膳立てし、完璧な指示（プロンプト）を与えなければ、望む答えは返ってこない「待ちの姿勢」であり、もし情報不足やエラーがあれば、そこで思考停止して「分かりません」と答えることしかできませんでした。

対して、２枚目の有機的なネットワーク図が「AI Agent」です。

ここでLLMが担っているのは、単なる回答作成機ではなく、システム全体を統括する「司令塔」としての役割です。

最大の違いは、一度の指示で終わるのではなく、「推論ループ」と呼ばれる試行錯誤のプロセスを持っている点です。

ユーザーから「来週の京都旅行のプランを」というゴールだけを渡されたとき、エージェントは即座に回答を出そうとはせず、まず人間のように「思考」を開始します。

❶LLMがユーザーからの依頼を受けて全体の計画を立てる：「まずは天気を調べ、次に新幹線の空き状況を確認し、その後にホテルを探そう」と段取りを組みます。

❷LLMが指示して何かしらのツールアクションを実行する行動：MCP（手足）を通じて、Web検索や予約サイトへのアクセスを実行します。

❸LLM自身による評価：もし「希望のホテルが満室」という結果が返ってきたら、「では、少しエリアを広げて再検索しよう」と自ら軌道修正を行います。

このように、外部環境からのフィードバックを受けて、ゴールに辿り着くまで何度でも行動と評価を繰り返す。

この「粘り強さ」と「自律的な修正能力」が、AI Agentが単なるプログラムを超えて、私たちが実務を任せられる「パートナー」へと進化した大きな技術的要因です。

❹A2A (Agent to Agent)：AI同士が協力する組織へ

そしてAI活用の未来形は、1つの万能なAIがすべてを行うのではなく、専門特化した複数のAIエージェントがチームを組むA2A（Agent to Agent）の世界へと向かいます。

「リサーチが得意なエージェント」が情報を集め、「分析が得意なエージェント」がそれを処理し、「文章作成が得意なエージェント」がレポートにまとめ、「監査エージェント」が内容をチェックする。

それぞれの得意分野を持つエージェント同士が、メッセージバスや共有メモリを通じて連携・協調し、人間一人では到底処理しきれない複雑で大規模なタスクを解決していくのです。

定義

特定のタスクに特化した複数のAIエージェントが、互いにメッセージをやり取りしながら連携・協調し、より複雑で大規模な課題を解決する仕組みです。

非技術者向けのイメージ

「何でもできるスーパーマン1人」に任せるのではなく、「各分野の専門家が集まるプロジェクトチーム」を結成して仕事に取り組むスタイルです。

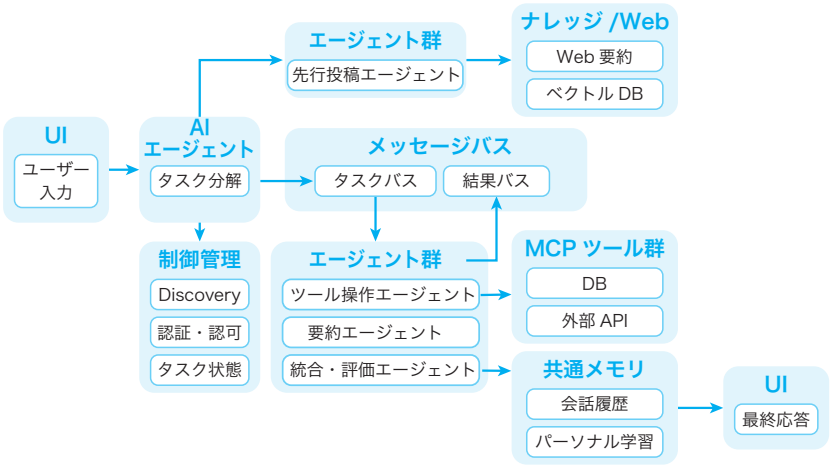
技術のポイント

1つのAI（シングルエージェント）ですべてを行おうとすると、扱える情報量（コンテキストウィンドウ）や能力に限界が来ます。A2Aでは、「調査担当」「執筆担当」「レビュー担当」のように役割を分担させることで、互いのミスをチェックしたり、並行作業で効率を上げたりすることが可能になります。

応用例

ソフトウェア開発において、仕様を決めるPM役、コードを書くエンジニア役、バグを探すテスター役のAIたちが議論しながらアプリを完成させる。

＊ A2A（Agent to Agent）の自律分散型アーキテクチャ



この図は、複数のAIエージェントが連携して複雑なタスクを解決する「A2A」の全体像を示しています。

中央の「メッセージバス」を介して、多様な専門エージェント（先行投稿Agent、ツール操作Agent、要約Agent、統合・評価Agent）が有機的につながっています。

ユーザー（UI）から指示を受けた制御管理エージェント（左下）は、まるで人間のプロジェクトマネージャーのように、タスクを分解して各エージェントに割り振ります。

それぞれのエージェントは、MCPツール群やナレッジ/Web（右上）を駆使して自分の担当領域を処理し、その結果を共有メモリ（右下）やメッセージバスに戻します。

こうして、個々のAIの能力を超えた「集合知」によって、大規模で複雑なゴールが達成されていく様子が見て取れます。

人間は、これらエージェントチームの「総監督（マネージャー）」として、目標を設定し、最終的な成果物を評価する役割へとシフトしていきます。生成AIの進化は、私たちに「プレイヤー」から「指揮者」への変化を促しているのです。

＊ LLMとRAG、MCP、A2Aの比較

比較観点	LLM	RAG	MCP	A2A（Agent to Agent）
入力・指示	人間の1回のプロンプト	質問 ＋ 検索クエリ生成	指示 ＋ 利用可能ツール定義	目標・制約・役割を宣言
知識源	モデルの事前学習知識のみ	ベクトル検索/構造化検索	外部DB・SaaS・ファイル群	共通メモリ/ブラックボード
自律性役割分担	なし（単一思考）	低～中（取得→生成の直列）	中（モデルが計画して実行）	高（計画/実験/評価の分業でモデルが合議）
オーケストレーション	不要	取得フローの簡易制御	ツール選択/再試行/停止条件実行ログ管理	メッセージ/タスク/結果バス共通メモリで合意を形成
代表ユースケース	要約・Q&A・創作・下書き	社内ナレッジ検索	データ取得・SaaS操作	複雑なワークフロー・意思決定・品質担保
強み	速い/安い/セットアップ最小	最新情報に強く、根拠提示可	実世界のアクションが可能	並列/分業で失敗を相互に検知・補正
注意点	幻覚、ドメイン弱	検索品質・インデックス整備が鍵	誤操作/権限管理・監査が必須	設計が複雑・観測/評価が必要

2-2 ローカルAIとパーソナルAI 自分だけの最強コックピット

前節では、企業や社会システムを変革する大規模な「AI Agent」の話をしました。しかし、AI Agentの恩恵を最もダイレクトに、そして今日から受けられるのは、私たち個人の「ローカル環境（PCの中）」です。

AIは今、「クラウドの向こう側にある賢いチャットボット」から、「自分のPCの中で一緒に働く同僚」へと進化しつつあります。Obsidianのようなナレッジ管理ツールと、CursorやAntigravityといったAIネイティブなエディタ（IDE）が融合することで、私たちのPC自体が「巨大なAIの脳」へと変わり始めています。

なぜ今、「ローカル」なのか？：情報のサイロ化解消

これまで、AIを使うためにブラウザを開き、ChatGPTにテキストをコピー＆ペーストしていた作業は、コンテキスト（文脈）の分断を生んでいました。AIはあなたのPCの中にあるファイルを知らないため、毎回ゼロから状況を説明する必要があったのです。

しかし最新のAI IDEは、ローカルにある膨大なMarkdownファイル（メモ、日報、原稿の断片）を「プロジェクト全体」として読み込みます。これにより、AIは単に命令されたことだけでなく、過去のあなたの思考プロセス、独自の用語定義、文体の好みまでを「ローカルファイル」から学習し、コンテキストとして理解した上で作業を代行できるようになります。

セキュリティリスクの高いクラウドにデータを預けることなく、自分に完全に最適化された「パーソナルAI」が、手元で高速に動く。これが、AI Agent時代の新しい働き方のスタンダードになりつつあります。

技術的革新が生む3つの変化

この「ローカルAI環境」は、具体的に私たちの作業をどう変えるのでしょうか。3つの革新的なポイントと、すぐに使える実践的なユースケースを紹介します。

革新ポイント①：ファイルを縦横無尽に横断してコンテキストを与えることができる

これまでは「資料を見る画面」と「AIに指示する画面」が別々でした。しかしAI IDEでは、今開いているファイルだけでなく、ユーザーがファイル単位やファイル内

の行単位で指定して、その内容をAIが自動で参照して回答絵を生成します。つまり、プロンプトは不要で、「あのファイルのあれ」と言うだけで通じる世界です。

革新ポイント②：Markdownプロトコルによる構造化

人間にとっても読みやすく、AIにとっても処理しやすい「Markdown」が共通言語になります。WordやExcelのような独自形式ではなく、プレーンテキストベースのMarkdownで情報を管理することで、AIはファイルの中身を正確に理解し、加工・出力が可能になります。

革新ポイント③：インライン・アシストによるフロー体験

チャット画面に移動することなく、エディタのカーソル位置でそのままAIを呼び出し、文章を生成・修正・変換できる機能です。思考を中断させず、呼吸するようにAIの力を借りることができます。

実践事例 コミュニケーション情報の「ストック化」と民主化

これら3つの革新性、①：全ファイル横断コンテキスト、②：Markdownによる構造化、③：インライン・アシスト　が組み合わさることで、これまでプロジェクトマネージャー（PM）の頭の中や個人のメモ、あるいは流れるチャットの中に埋もれていた情報は、チーム全員が活用できる「共有知」へと進化します。

ここでは、会議議事録やチャットログを起点とした「プロジェクト情報の民主化」を例に、6つのステップに細分化した新しいワークフローを紹介します。

flow.1 フロー情報の構造化（Markdown化）

会議の音声やチャットのログといった「流れ去る情報（フロー）」を、AIが理解・処理しやすいMarkdown形式の「蓄積される情報（ストック）」に変換します。

A. 会議音声の場合

ミーティングの音声を文字起こしし、構造化された議事録へ変換します。

- 情報のワークフロー ［会議音声］→［文字起こしテキスト］→［LLMによる構造化］→［Markdownファイル(.md)］
- LLMの役割 文字起こしされたテキストを入力とし、事前に定義されたシステムプロンプト（「決定事項」「ToDo」「期限」を抽出せよ等の指示）に基づいて、構造化された議事録を生成します。
- 具体例 1時間のZoomミーティングのログから、ToDoを抽出しチェックボックス付きリストとしてファイルに書き込みます。

B. Slack等のチャットログの場合

指定したチャンネルやスレッドの議論を回収し、要約・Markdown化します。

- 情報のワークフロー [Slackチャンネル/スレッド指定] → [ログ抽出] → [LLMによる要約・構造化] → [Markdownファイル(.md)]
- LLMの役割 雑多な会話から「議論の背景」「決定プロセス」「共有リンク」を抽出し、文脈がわかるレポート形式にまとめます。
- 具体例 「project-alpha」チャンネルの直近1週間のログを解析し、重要な議論の流れだけを要約した `weekly-digest.md` を生成して保存します。

flow.2 クラウドとローカルの「知の同期」

生成されたMarkdownファイル群（議事録やチャット要約）をクラウドストレージに保存し、それをチーム全員のローカル環境に同期させます。

- 情報のワークフロー [Markdownファイル群] → [SharePoint/Google Drive] → [各メンバーのPC内ローカルフォルダ]
- 具体例 プロジェクトメンバーのPCには、常に最新の議事録とチャットログがフォルダとして存在している状態を作ります。

flow.3 ローカルAIエージェントへの「コンテキスト注入」

メンバー個々人が使うAI IDE（Cursorなど）に、同期されたフォルダ全体を読み込ませます。

- LLMの役割 単一のファイルだけでなく、過去の議事録やチャットログすべてを「長期記憶」として保持し、プロジェクトの経緯や文脈を深く理解します。
- 具体例 新しくチームに参加したメンバーがフォルダをAI IDEに読み込ませるだけで、AIはそのプロジェクトの全歴史を把握した「専属秘書」になります。

flow.4 自然言語による情報の抽出（Q&A）

メンバーはPMに質問したりSlackを検索する代わりに、エディタ上のAIエージェントに質問します。

- 情報のワークフロー [メンバーの質問] → [AIエージェント（ローカル検索）] → [回答]
- 具体例 例1）「先週のSlackで議論されていた、デザイン修正の結論はどうなった？」
  例2）「直近の会議で決まったAさんのタスク期限はいつ？」

AIはローカルファイルを横断検索し、会議とチャット両方の文脈を統合して正確な回答を返します。

flow.5 AIエージェントによる自律的な更新

情報の閲覧だけでなく、修正や更新もAIエージェントを介して行います。

- LLMの役割 ユーザーの指示に基づき、該当するMarkdownファイルを直接編集します。

- **具体例** エディタ上で「このタスクは完了したのでステータスを更新して」と指示すると、AIが議事録ファイルの該当箇所を書き換えます。

flow.6 「Gitライク」なバージョン管理と共有

ローカルで更新された情報は、クラウドを通じて再びチーム全体に共有されます。

- **情報のワークフロー** [ローカルでの修正] → [クラウド同期（差分表示）] → [チーム全体へ反映]
- **具体例** ソフトウェア開発の「Git」のように、誰が・いつ・何を書き換えたかが履歴として残ります。これにより、PMの属人的な管理コストは大幅に削減され、情報の透明度が飛躍的に向上します。

このように、ローカルファイル（Markdown）を共通言語とし、AIエージェントを介在させることで、情報の「入力・蓄積・活用・更新」というサイクルそのものを自律的かつ透明性の高いものへと変革できます。また、仕事の失敗の多くは人間同士のコミュニケーションエラーなので、できるだけ人間との会話を減らして、LLMとの対話を増やした方がプロジェクトはうまくいくのではないかと考えています。

ファクトベースのコミュニケーションはLLMを中心にして、人間同士の会話は、キャリアの相談、人間関係の不和調整、士気を上げるリーダーシップなどのピープルマネジメントに集約していくのが私個人の解でもあります。

2-3 10年後のAIリテラシー

AI Agentが驚異的なスピードで進化し、多くの知的作業を代行してくれるようになったとき、私たち人間に求められるスキルはどう変わるのでしょうか。

前回のコラムで、「知能（合理的な正解を出す力）」はAIに任せ、人間は「知性（意味づけを行う力）」を磨くべきだという話をしました。では、この抽象的な「知性」を、AI時代の実務における具体的な「スキル」として翻訳すると、どのような形になるのでしょうか。

どんなにAIが進化しても陳腐化せず、むしろAIが賢くなればなるほど重要性を増す、3つの普遍的なスキルがあります。

❶AIコンテキストの設計力（文脈を授ける力）

コラム1で触れたように、これからのAIは単なるツールではなく「バディ（相棒）」になります。優秀なバディに最高の仕事をしてもらうために必要なのは、細かい命令ではなく「文脈（コンテキスト）の共有」です。

「メールの返信を書いて」という指示だけでは、AIは一般的な（つまり退屈な）文面しか作れません。「相手は長年の付き合いがある〇〇さんで、前回少し無理をお願いしてしまった経緯がある。だから今回は、感謝の気持ちを厚めにしつつ、やんわりとスケジュールの調整を依頼したい」といった、人間関係の機微や暗黙の了解を含めた「文脈」を言語化し、AIにインプットする力。

これこそが、AIの「知能」を最大限に引き出すための、人間にしかできない「知性」の行使です。単なるプロンプトエンジニアリングを超えた、AIへの「意味の伝達能力」が問われます。

❷ドメイン知識に基づく「審美眼」（価値を決める力）

AIは、私たちが投げかけた問いに対して、平均点以上の「それっぽい正解」を瞬時にいくつも提示してくれます。しかし、その中から「どれが本当に優れているか」「どれが今の文脈で最も人の心を動かすか」を選ぶのは、人間にしかできません。

ここで重要になるのが、コラム1で述べた「リベラルアーツ（教養）」や、特定の分野（ドメイン）に対する深い知識です。

例えば、AIが生成した10個のデザイン案の中から、ブランドの歴史的背景や現代の空気感に合致する「最高の一つ」を選び取る審美眼。あるいは、AIが書いたコード

に対して、保守性やセキュリティの観点から「採用可否」を判断する専門知識。

AIという「優秀な部下」が持ってきた成果物に対し、責任を持ってGoサインを出し、あるいは修正を指示する「ディレクターとしての目利き力」が、これからの専門性の正体となります。

❸デジタルツインの育成力（自分をデータ化する力）

前節で触れた「ローカルデータマネジメント」の発展形です。あなたの思考、経験、価値観といった「知性の源泉」を、AIが理解できる形のデータとして蓄積し続ける力です。

日々のメモ、読んだ本の感想、会話のログ、過去の成果物。これらを整理し、AI(バディ）がいつでも参照できるようにしておくこと。それは言わば、デジタル空間にもう一人の自分（デジタルツイン）を育てるようなものです。

AIはあなたのローカルデータという「燃料」を与えられて初めて、一般的な正解ではなく、「あなたらしい正解」を出せるようになります。自分自身の知的資産をマネジメントし、AIというエンジンの性能を極限まで高められる人が、これからの時代に圧倒的なパフォーマンスを発揮することになるでしょう。

2-4 AIによって天才は「民主化」される？

かつて、圧倒的な成果を出す「天才」たちの仕事は、凡人には理解できないブラックボックスの中にありました。

「なぜかあの人は正解にたどり着く」「あの人の直感は外れない」。その背景には、膨大な知識量に裏打ちされた高速な論理的思考や、経験則に基づく独自のアルゴリズムがあったはずです。しかし、それらは「センス」や「才能」という曖昧な言葉で片付けられ、他人には再現不可能なものとされてきました。

しかし、LLM(大規模言語モデル）の登場は、この「天才の専売特許」を一般層へと開放しつつあります。これが「知能の民主化」です。

具体的に何が起きているのでしょうか。

第一に、「暗黙知の形式知化」です。

これまで天才の頭の中にしかなかった高度な思考プロセスが、AIとの対話を通じて言語化され、誰もが使える「プロンプト」や「ワークフロー」として共有可能になります。例えば、熟練プログラマーが数時間かけて行っていたデバッグや設計の勘所も、AIが瞬時に提示してくれるようになります。

第二に、「スキルの底上げ（フロア・レイジング）」です。

画像生成AIを使えば、絵筆を握ったことのない人がプロ級のイラストを描けるように、LLMを使えば、高度な論理的思考や多角的な分析、美しい文章作成といったスキルが、誰にでも「標準装備」されます。

かつては習得に10年かかったスキルが、AIという「補助輪」ではなく「高性能エンジン」を搭載することで、一瞬にして手に入るのです。

これは、天才の価値がなくなることを意味しません。むしろ、「天才の定義」が変わると言えるでしょう。これまでの天才が「誰も解けないパズルを解く人（Howの天才）」だったとすれば、これからの天才は「誰も思いつかない新しいパズルを作る人（Whatの天才）」へとシフトしていきます。

既存の解法で解ける問題はAIと、AIを使いこなす「拡張された個人」が解決してしまうからです。卓越した才能がコモディティ化し、誰もがかつての「天才レベル」の成果を出せるようになる。

そうして人類全体の平均値が劇的に底上げされた世界で、私たちは「自分だけの問い」を見つけ、それを解くためにAIと共に歩んでいくことになります。