CDATA

調査レポート：エグゼクティブサマリー

Claude Code に企業向け MCP サーバーを作らせてみました。分かったことを報告します。

企業のデータアクセスに向けた、AI 支援による
MCP サーバー開発の評価

2026 年 7 月（日本語版）

CDATA

問い

Model Context Protocol（MCP）は、企業のデータアクセスに新しいカテゴリーを切り開きました。AI エージェントは今や業務システムに直接つながり、リアルタイムのレコードに問い合わせてアクションを実行し、古びたエクスポートではなく企業の現在の状態を映した答えを返せます。同時に、Claude Code や GitHub Copilot のような AI コーディングツールの登場で、ソフトウェア開発はかつてないほど速く、コストを抑えて行えるようになりました。多くのタスクでは、最初に出てきた成果物がほとんど手直しなしで本番に投入できる水準に達しています。AI エージェントを導入する企業にとって、この組み合わせは魅力的なものです。エージェントが到達すべきデータソースにはそれぞれコネクタが必要で、AI コーディングツールがそのコネクタを本番に耐える形で作れるなら、AI によるデータアクセスの経済性とスピードは大きく変わってくるでしょう。ただし、それが実際に可能かどうかは、まだはっきりしません。

AI コーディングツールで、信頼できて企業で本番運用できる水準の、本番に投入できる MCP コネクタを作れるのか。私たちはそれを確かめることにしました。

私たちは 2 つのアプローチで、合わせて 9 回のセッションを実施しました。1 つ目は、多くの開発者が最初に取る方法を再現するため、最小限のプロンプトだけを与えた素の Claude Code を使うアプローチです。2 つ目は、データ連携の経験を持つ開発者が加わり、構造化されたワークフローとベストプラクティスを適用するアプローチです。そこで分かったことを、以下に詳しくまとめます。

基準

最初のステップは、実際の企業環境で信頼されるために、コネクタが満たすべき条件を定義することでした。私たちはこの基準を、6 つの観点にわたって定めました。

正確性と網羅性。データソースにある内容と一致する、正確なデータを返します。正しいレコード、正しいフィールド、正しい値です。フィルタリング、並べ替え、日付ロジックは想定どおりに動作し、黙って打ち切られることもありません。

セマンティックな名前解決。データが保存されている形と、ユーザーや AI が自然に理解する形との間にある溝を埋めます。フィールド名、解決済みの関係性、ビジネスロジックのいずれもが、内部表現ではなくデータソースのシステムが持つ本来の意味を映し出します。

大規模データでの耐障害性。15 行でも 4 万行でも、一貫した挙動を保ちます。ページネーション、レート制限、トークンの失効、一時的なエラーのいずれも、エラーを出さないまま落ちたり人手を介したりすることなく処理します。

認可の忠実な再現。ユーザーには、その人が見る権利を持つ範囲だけが見えます。共有のサービスアカウントを介してではなく、ユーザー一人ひとりの単位で権限を効かせます。

機能の深さ。データソースには存在するものの、標準スキーマには含まれないカスタムフィールド、カスタムオブジェクト、カスタムエンティティにも対応します。

変化への適応性。API、認証フロー、フィールドの型など、データソースのシステム側が変わっても、コネクタは全面的な書き直しを必要とせずに追従します。

これらのどれか一つでも欠けたときの代償は、クエリが 1 件失敗する程度では済みません。AI エージェントが不完全なデータを何も告げずに返し、古いレコードにもとづいて事業判断が下され、エンジニアリングチームが数週間後にその問題に気づく――そういう事態につながります。

基準を定めたところで、それを公正に試すデータソースが必要になりました。私たちが選んだのは SharePoint です。企業でもっとも広く導入されているプラットフォームの一つであり、プロジェクトトラッカー、購買記録、人事関連のリスト、コンプライアンス台帳、契約データといった構造化された業務データが大量に存在する場所でもあります。加えて、その複雑さはよく知られています。認証まわりの要件、Microsoft Graph に特有のページネーションの挙動、ルックアップの関係、そしてドキュメントが示唆する内容とは異なるスキーマの慣習です。SharePoint を本番環境で正しく扱えるコネクタなら、その名に値するといえます。

SharePoint 向けの MCP サーバーがきちんと動けば、AI エージェントは次のようなプロンプトに答えられるようになります。

- 「今四半期に提出された、$50,000 超の未処理の購買リクエストをすべて見せて」
- 「直近 30 日以内に入社した社員のうち、オンボーディング記録がまだ未完了なのは誰か」
- 「年内に満了する契約をすべてリストアップし、更新担当者が割り当てられていないものにフラグを立てて」

これらは SharePoint のリストに対する構造化データへの問い合わせであり、業務部門が日常的に尋ね、いまはエクスポートを取り出すか IT 部門を待つかして答えている類いの質問です。信頼できる MCP サーバーがあれば、これらの質問にリアルタイムで、しかもユーザーごとの権限を効かせたまま答えられるようになります。

評価軸

私たちは、上記の基準に照らして検証するための評価軸を 8 つ定めました。それぞれが、実際の企業導入で表面化する失敗のパターンに対応しています。概念実証（PoC）の段階では現れず、業務部門のユーザーが大規模なエクスポートを実行したとき、2 つのリストにまたがる質問を投げたとき、あるいは AI エージェント経由でこれまで一度もアクセスしたことのないシステムに問い合わせたときに初めて何かが壊れる――そういう類いの失敗です。この 8 つを合わせると、デモでは動くコネクタと、企業が頼りにできるコネクタとの違いが浮かび上がります。

- **OAuth トークンのライフサイクル**――複数ページにわたるデータ取得の途中で、トークンの失効に対応できるか（大規模データでの耐障害性、認可の忠実な再現）
- **フィールド取得の方式**――フィールドを静的に定義するのではなく、API から動的に取得しているか（正確性と網羅性、機能の深さ）

- **フィールド名のわかりやすさ**——返されるフィールド名が、生の API 内部名ではなく SharePoint の UI 上のラベルと一致しているか（セマンティックな名前解決）
- **ツールパラメータの設計**——内部識別子を事前に知らなくても、ユーザーが SharePoint のリストに問い合わせられるか（正確性と網羅性）
- **リストのリレーション処理**——ルックアップカラムやリスト間参照が、意味のある値まで解決されているか（セマンティックな名前解決、機能の深さ）
- **ページネーションの正確性**——Microsoft Graph に適した方式でページネーションが実装されているか（大規模データでの耐障害性、正確性と網羅性）
- **大規模データセットでの挙動**——リストやライブラリが大半のエンドポイントの最大ページサイズを超えたときも、一貫した挙動を保てるか（大規模データでの耐障害性）
- **エラー処理と耐障害性**——SDK の制約、タイムアウト、大きなペイロードを、エラーを出さないまま落ちることなく処理できるか（大規模データでの耐障害性、変化への適応性）

主な調査結果

動くだけのスキャフォールディングと、何千件ものレコード、失効していくトークン、ルックアップの関係、そして規模が大きくなって初めて表面化するプラットフォーム固有の挙動まで扱えるコネクタとの間には、大きな違いがあります。Claude Code は最初のハードルは越えました。しかし 2 つ目のハードルは越えられませんでした。

素の Claude Code では、8 つの評価軸のうち、人の手を介さずに合格したのは 1 つだけでした。経験豊富な開発者が Claude Code のベストプラクティスを適用し、追加のセッションを何度重ねても、3 つの軸は最後まで完全には解決しませんでした。もっとも重大な不具合を修正するためのデバッグ作業は、それまで正常に動いていたコンポーネントを、意図せず壊してしまいました。

これは、プロンプトの与え方が悪かったことによる結果ではありません。最初のプロンプトは、多くの開発者がこの課題にどう取り組むかを再現するため、意図的に最小限にとどめました。専門家の指導ありのフェーズでは、Claude Code に最善を尽くす機会を与えています。それでも生じた失敗は、もっと根本的な何か——コンパイルが通るコードと、企業の環境で持ちこたえるコードとの間にある距離を映し出しています。

合格とは、人の介入なしで正しく実装されていることを指します。一部合格は、動作はするものの、実運用での検証で不足が見つかる状態です。不合格は、実装が誤っているか存在せず、人による原因調査と書き直しが必要な状態を指します。

評価軸	素の Claude Code	専門家の指導あり	実運用への影響
OAuth トークンのライフサイクル	不合格	一部合格	コンプライアンス担当者が 25,000 件をエクスポート。360 ページ目で、警告もなく処理が止まります。画面上は完了。実際には完了していません。
フィールド取得の方式	合格	合格	業務部門が追加したカスタムカラムも、そのまま自動で反映。スキーマが変わっても、作り直しは要りません。
フィールド名のわかりやすさ	一部合格	一部合格（改善）	AI の答えは AuthorLookupId: 83。これを渡されたユーザーに、できることはありません。データはそこにあるのに、使い物にならないのです。
ツールパラメータの設計	一部合格	一部合格（改善）	見慣れないリストにクエリを投げるたびに、ユーザーが見たこともない識別子を IT 部門が用意することになります。MCP の売りだったセルフサービス性は、そこで消えます。
リストのリレーション処理	不合格	一部合格	取引先と請求書をまたぐ質問に、返ってくるのは生の ID だけ。エージェントは答えを出せません。ユーザーは結局、手作業でレポートを取り出すところに戻ります。
ページネーションの正確性	不合格	合格	6,000 件あるリストが返すのは 5,000 件。警告は、ありません。欠けたデータのまま意思決定が進みます。
大規模データセットでの挙動	不合格	合格	12,000 件の資産レコードのエクスポートが、警告もなく 10,000 件で止まります。欠落が見つかるのは数週間後、ハードウェア監査の場です。
エラー処理と耐障害性	不合格	一部合格	15 件取るだけのクエリが 5 分間応答せず、結果はゼロ。ユーザーは、二度とそのツールを開きません。

失敗パターンの分析

失敗はでたらめに起きたものではありません。ほとんどは 4 つのパターンに集約され、そのどれもが同じ根本的なずれに行き着きます。AI コーディングエージェントは「動くソフトウェア」を作ることに最適化されています。しかし企業の本番環境が求めるのは、「動き続けるソフトウェア」です。しかも、最初の実装が想定していなかった条件のもとで。

ハッピーパス最適化。Claude Code が生成したコードは、小規模で単純なシナリオでは動作するものの、規模が大きくなると破綻します。初期実装は 3 件では動きましたが、15 件でハングしました。ページネーションの実装はなく、安全装置の上限値も実際の閾値を踏まえずに設定されていました。評価軸 8 つ中 5 つで見られたパターンです。

無批判な依存関係の選定。SDK のバージョンは、既知のバグや未解決の Issue、バージョン固有の制限を確認しないまま選ばれていました。最初に見つかった使えそうな依存関係を、実運用での実績やチェンジログの履歴と照らし合わせることなくそのまま採用しています。最も重大な失敗の根本原因です。

プラットフォームに関する誤った前提。Microsoft Graph 特有の挙動が必要な場面に、汎用的な REST のパターンをそのまま当てはめていました。SharePoint がカーソル方式の @odata.nextLink を要求するところで、スキップ方式のページネーションを前提にしていたのです。対象リストを検出する処理が要件として想定されていないケースもありました。評価軸 8 つ中 3 つで見られたパターンです。

ライフサイクルへの意識の欠如。すべての API 呼び出しを、それぞれ独立したものとして扱っていました。ページネーションの呼び出しをまたぐトークンの失効、再帰的な走査を重ねるごとに膨らむレスポンスサイズ、そしてペイロード上限とリトライ挙動が時間とともに積み重なる複合的な影響――いずれにも対処していません。評価軸 8 つ中 3 つで見られたパターンです。

Claude Code で十分な場面もある

今回の評価は、AI 支援によるコネクタ開発そのものを一律に否定するものではありません。Claude Code が実際に開発を加速できる場面もあります。

PoC での検証。Claude Code は、本格的な実装に踏み切る前に接続の実現可能性を示すのに向いています。アーキテクチャは妥当な出発点であり、小規模なデータセットならすぐに動きます。これは確かな価値です。とりわけ、内製か購入かを判断する前に実現可能性を確かめておきたいチームにとってはなおさらでしょう。

経験あるコネクタ開発者にとってのスキャフォールディング。企業環境特有のエッジケースがどこに潜んでいるかを知っている開発者の手にかかれば、Claude Code は開発初期の時間を圧縮してくれます。しかも、そのエッジケースを知らない場合に表面化するリスクを持ち込みません。ボイラープレートは正しく、構造もきれいです。抜け落ちている部分も、探すべき場所を知っていれば予測できます。

十分と言えないケースは、まさに本番の AI ワークロードで重要になる場面そのものです。実際のユーザー、実際のデータ量、複雑なスキーマとリレーションモデルを持つ企業のデータソース、そして複数のエンティティにまたがる質問に正しく答えなければならない AI エージェント――これらが絡む場面です。

なお、今回の評価は Claude Code に限定したものであり、検証対象も SharePoint に絞られている点には留意が必要です。他の AI コーディングツールも急速に進化しており、性能特性はツールごとに異なります。あるツールについて今日成り立つことが、別のツールでは明日には成り立たないかもしれません。

CDATA

本番運用に耐えるコネクティビティが本当に必要とするもの

企業環境で動くコネクタと、本番で持ちこたえるコネクタとの間には違いがあります。企業環境はデモを乗り切らせてくれます。本番とは、コンプライアンス担当者が午後 11 時に 25,000 行のエクスポートを実行するときや、調達に関する問い合わせが 3 つのリストにまたがり、8 ページ目でトークンが失効するときに実際に起きることです。

想定していない事態に対応する

処理の途中でのトークン失効。データソースのシステム側で起きる API の変更。顧客が最大規模のデータセットを送ってきたときにしか現れない、ページネーションのエッジケース。エラーを出さないまま失敗し、下流で何かがおかしくなるまで気づかれないこと。これらは例外的なシナリオではなく、企業のソフトウェアが置かれる通常の運用条件そのものです。

データソースのシステムへのセマンティックな理解を備えている

ユーザーが UI で目にする表記と一致するフィールド名。意味のある値に解決されるルックアップのリレーション。ドキュメントが示唆する動作ではなく、プラットフォームが実際にどう振る舞うかを反映したビジネスロジック。こうした知識は、API を一度読んだだけでは得られません。実装を重ね、失敗のたびに磨かれることで蓄積されていくものです。

機能の深さに対応する

標準スキーマに含まれないカスタムフィールド、カスタムオブジェクト、カスタムエンティティ。読み取りにとどまらず、データソースのシステムの操作レイヤーにまで踏み込むアクション。API のドキュメントに書かれている範囲を扱えるコネクタと、本番で API が実際にどう振る舞うかまで扱えるコネクタは、同じではありません。

作って終わりではなく、保守され続けている

API は変わります。認証フローも変化します。レート制限も変わっていきます。本番運用に耐えるコネクタは、データソースのシステムが変化したときに、全面的な書き直しを必要とせず適応します。エンジニアリングのコストは開発初期に集中しているわけではありません。コネクタの寿命全体で見れば、保守が全体の労力の大半を占めます。

結論

Claude Code は、自然言語の指示だけから、正しい認証フロー、明快な関心の分離、適切な MCP プロトコル処理を備えた、構造の整った Java アプリケーションを作り上げました。アーキテクチャは健全で、コードの品質もおおむね高い水準にありました。小規模なデータセットを扱う PoC のシナリオであれば、最初の出力がそのまま動きました。

しかし、企業環境で使われる SharePoint は PoC のシナリオではありません。リストには数千件のアイテムが含まれます。フィールドは UI 上のラベルとは異なる API 内部名を使っています。ルックアップカラムは複数のリストにまたがるリレーションを作り出します。長時間の処理が続く間にトークンは失効します。SDK のデシリアライズ層は、規模が大きくなると何も告げずに失敗します。

人の手を介さずに合格した評価軸は 1 つだけでした。専門家の指導があっても、一部合格のまま残った評価軸が 3 つありました。最も重大な失敗を修正した際には、認証が壊れてしまいました。しかもその失敗の根本原因は、既知で修正可能な SDK のバグであり、Claude Code は単体テストの中では正しく特定していました。それでも、提案された修正は最適なものではありませんでした。

問うべきは、Claude Code がコネクタを書けるかどうかではありません。書けます。問うべきは、ユーザーがそれに頼る場面で持ちこたえるかどうかです。今回の評価——2 つのフェーズ、9 セッション、あらゆる段階に経験豊富な開発者が関わった——を踏まえる限り、まだ簡単に答えが出る問いではありません。

本番運用を見据えて作る

CData Connect AI は、本評価が検証対象とした MCP 接続基盤そのものです。本番の AI ワークロード向けに MCP 接続を検討しているチームにとって、Connect AI はここで評価した各評価軸をカバーしています。実運用で規模が拡大したときにしか表面化しない挙動まで含めてです。

エッジケースへの対応も含め、Claude Code が生成したコネクタと CData の SharePoint コネクタを直接比較した内容は、比較分析レポートをご覧ください。