

AI時代の開発職は なぜ変わるのか

利用者の事実と背景から
設計・検証・採用・育成を考える

何を作れるかに加えて、
誰の仕事が、どう進むと良いのか。

その理由を確かめ、
実際に使える仕組みにする。
開発の役割を、ここから考えます。

リクエスト株式会社 甲畑 智康

組織行動科学® | 2026年10月5日

前作「AI時代の開発職は、何を担うのか」の続編

リクエスト株式会社 | 組織行動科学®



理由から実際の開発へつなぐ

ソフトウェアの価値は、使う人が仕事を進められるかによって変わります。本資料では、この考え方を開発の判断に落とし込みます。

3-8 そもそもなぜ利用者の事実と背景が重要なのか

9-16 備品修理の受付を例に設計・検証・運用を考える

17-20 開発職の採用と若手育成を考え直す

21 一件の開発で使う記入用書式

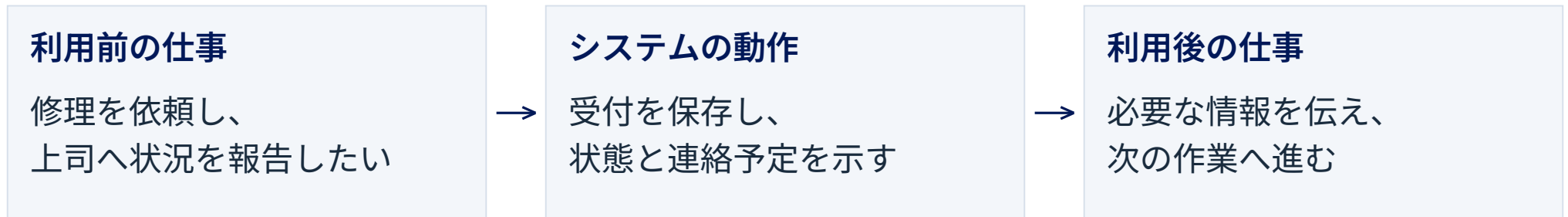
22-24 参照した一次資料12件

対象は、Web・ソフトウェア開発者と、開発の依頼・採用・育成を担う方です。利用者理解はAI以前から重要であり、ここではAIによる仕事の配分の変化と結び付けて考えます。

備品修理の場面・会話・仕様は架空例です。手順・書式・選考案は実践提案であり、効果を検証済みの評価尺度ではありません。

良し悪しは使う人の仕事の中で決まる

プログラムが正しく動くことは重要です。その動きが誰の何を助けるかは、利用される場面まで確かめる必要があります。



例えば、備品修理の依頼を送信できても、受付済みか分からなければ、本人は確認の電話をかけるかもしれません。送信機能が動くことと、本人が安心して次の仕事へ移れることには、確かめる対象の違いがあります。

開発では、操作の完了と、その後の仕事の進み方を結び付けて考えます。必要な情報や次の判断が分かると、作るべき機能も具体的になります。

最初に確かめる問いは、この変更で誰が何を進められるようになるか。

架空例に基づく本資料の整理。利用者の目的とサービス全体の評価を参照。[1][3][10]

同じ行動でも必要な仕組みは変わる

備品修理の依頼後に、二人が同じ確認の電話をかけたとします。行動の回数だけでは、必要な改善を決められません。

本人に確かめた背景	次に進めたい仕事	検討する仕組み
Aさん 依頼が届いたか分からない	受付済みと確かめ、 自分の作業へ戻りたい	受付番号と、 保存済みの受付状態を示す
Bさん 上司へ今日中に報告が必要	分かっている状況と、 次の回答時期を報告したい	現在の状態と、 次の連絡予定を示す

Bさんへ受付番号だけを知らせても、報告に必要な情報が足りない可能性があります。背景を知ると、同じ行動の中にある目的や制約の違いが見えてきます。

行動を減らす前に、その行動によって本人が何を確かめていたかを聞く。

説明のための架空例。二人の違いを、利用者全体の分類とみなさない。[1][2]

背景は判断に使っている条件から探る

背景を捉えるとは、性格を推測することではありません。本人の行動に関わる条件を探し、記録や対話で確かめることです。

見る範囲	確かめること	質問の例
前後の仕事	利用前のきっかけと 利用後に進める作業	この情報が分かると、 次に何ができますか
関わる相手	報告・承認・受け渡しの 相手と必要な情報	誰に、何を伝える 必要がありますか
時間と環境	締切、使う端末、 中断、支援の有無	いつまでに必要ですか。 普段はどう確かめますか
失敗の影響	間違えた場合に 止まる仕事と負担	分からないままだと、 誰の何が止まりますか

聞いた理由も、記録・実際の行動・別の場面と照らして確かめる。

質問は実践提案。支援を受ける人、利用を諦めた人、運用する人も観察の対象に含める。[1][2]

作る時間と確かめる時間の配分が変わる

AIは、実装案・コード・テスト案を作る仕事を助けます。一方、生成物を読み、現場の条件に合うかを確かめる仕事も生じます。

研究から分かること

DORAの2026年の報告は、Googleの開発者による自由回答1,110件を分析し、生成の速さと、確認・レビューの負担の両方を整理しています。自己報告による定性的研究で、全企業に共通する時間短縮率を測ったものではありません。[4]

一律に決められないこと

METRは2026年の追試報告で、参加者や課題の選られ方、並行作業の時間計測が、AIの効果の推定を難しくすると説明しています。過去の一つの効果量から、現在の全開発を予測することはできません。[5]

開発現場での意味

実装を速く進められる場面では、目的の確認やレビューが次の待ち時間になる可能性があります。自分のチームで、どの作業が速くなり、どこに負担が移ったかを確かめます。これは上記を踏まえた実践上の提案です。

AIの効果は、課題・経験・道具・組織の条件で変わる。特定の職種の消滅を実証した研究ではない。[4][5]

詳しい指示にも確かめるべき前提がある

AIに正確な仕様を渡しても、仕様の出発点が実際の仕事とずれていれば、期待した変化につながらないことがあります。

最初に受けた依頼

「確認の電話を減らしたい。受付後に自動通知を送ってほしい」

ここには、電話が何の確認なのか、誰が困っているのか、という前提が残っています。

背景を確かめた後

「受付の成否を知りたい人と、次の回答時期を知りたい人がいる」

通知の有無に加え、伝える情報、更新する人、情報が未確定のときの表現を決めます。

AIにも、確認済みの記録を渡して論点や案を整理してもらえます。その際、AIが補った説明を観察事実混ぜず、根拠と未確認の点を追えるようにします。

指示の詳しさに加え、その指示を必要とした事実を共有する。

架空例。AIへの具体的な作業範囲・文脈・受入条件の共有については公式文書を参照。[6]

使う人への約束を技術で成り立たせる

「受付済みと分かるようにする」という約束にも、保存、通信、権限、通知の設計が関わります。利用者理解と技術の判断はつながっています。

本人への約束	開発者が決めること	誤ると起こること
依頼が受け付けられたと確かめられる	保存成功を基準に、受付番号を返す	保存前の成功表示により、未受付を見落とす
通信失敗後も同じ依頼を確かめられる	同じ送信の再試行を同じ受付記録へ結び付ける	二重受付になり、別の担当者も動く
必要な人が自分の依頼を見られる	閲覧・更新の権限をサーバー側でも検証する	他人の依頼や担当者向け情報が見える

説明を受け入れる力に加え、動作とデータを追って確かめる力を育てる。

架空例の設計上の論点。再試行の扱いは[7]、リクエストごとの権限検証は[8]を参照。

誰の何が良くなるかを並べて考える

利用者、受付担当、手配担当では、必要な情報や負担が異なります。良い状態は、一人の便利さだけで決められません。

関わる人	実現したい状態	増やしたくない負担
依頼者	受付状態と次の連絡予定を確認して、必要な報告ができる	何度も入力する、 回答時期を推測する
受付担当	情報不足と未対応を把握し、 次の担当へ渡せる	複数の台帳を更新する、 重複依頼を探す
手配担当	判断に必要な情報が揃い、 進めるか差し戻すかを決められる	未確定の予定を約束する、 不要な通知を処理する

全員の希望をそのまま入れると、操作や運用が増えることもあります。今回は何を優先し、どの不便を残すかを業務責任者と決め、理由を残します。

架空例。AIは案の比較を支援できるが、現場の優先順位と採用する案の責任者を明確にする。[1][3]

実現したい状態を確認できる条件にする

今回の目標を「依頼者が、受付状態と次の連絡予定を確認し、必要な相手へ報告できる」と置きます。次の条件は、そのための仮の仕様です。

受付を確かめる

依頼を保存した後に受付番号を表示する。再読み込みしても、権限のある本人が同じ受付記録を確認できる。

次の行動を決める

状態、更新日時、次の連絡予定を表示する。予定が決まっていなければ未定と示し、問い合わせ先を載せる。受付済みを修理の手配済みと誤解させない。

更新を続けられる

状態と連絡予定を誰が更新するかを決める。既存台帳との二重入力を避ける方法も比較する。自動通知、画面表示、業務手順の変更から、小さく試す案を選ぶ。

仕様には、画面の項目と、その情報を正しく保つ仕事の両方を書く。

架空例に基づく実践提案。連絡予定は実際の担当体制で守れる条件を確認して決める。

受付と手配と通知を混同しない

依頼の保存、担当者の判断、通知の送信は別の出来事です。一つの「完了」にまとめると、本人も運用担当も状況を誤解しやすくなります。



通知の状態は別に記録する 送信待ち／送信済み／送信失敗

保存は成功し、通知だけが失敗する場合があります。このとき、依頼そのものを再登録させず、保存済みの受付記録を示し、通知の再送を扱います。

再試行では、同じ送信を識別する情報を使います。識別情報と保存を一体で扱い、同時に届いた場合も確認します。別の依頼まで重複として消さない条件が必要です。[7]

どこまで済み、何が未完了かを、画面・記録・運用で揃える。

図は架空業務の一部。差し戻し・取消などの状態、再試行の有効期間も実際の業務に合わせて設計する。

使う人への約束からテストを決める

生成されたコードの動きだけに合わせてテストを書くと、仕様そのものの抜けを見逃すことがあります。先に期待する結果を言葉にします。

確かめる場面	期待する結果	確かめ方
保存後に画面を開き直す	同じ受付番号と 現在の状態が見える	画面と保存記録を照合
保存後に応答が途切れ、 同じ送信を再試行する	受付は一件のまま。 同じ受付番号を返す	通信失敗と同時再試行を再現
他の依頼者として 記録へアクセスする	許可されていない情報を 返さず、更新もできない	画面とAPIの両方で確認
受付結果が画面内で変わる	読み上げ利用でも 結果を把握できる	支援技術で操作して確認

加えて本人に使ってもらい、受付と手配の違いを説明できるか確かめる。

APIはシステム間の呼出し口。この表は架空例の一部で、網羅的な試験一覧ではない。[7][8][9]

事実と判断と作業範囲を渡す

次は、備品修理の受付を試作する依頼文の例です。実案件では、許可された情報と、確認した実際のファイル名・実行方法へ置き換えます。

確認したことと未確認

依頼者は受付後に電話で状況を確認した。本人は「上司へ次の連絡時期を報告したい」と話した。他の依頼者にも同じ事情があるかは未確認。

目標と今回の範囲

受付状態と次の連絡予定を確認できる試作を作る。今回は検証用データを用い、実際の通知は送信しない。既存の認証・権限を保ち、保存成功前に受付済みと表示しない。

返してほしいもの

現行実装を読み、変更箇所と理由を示す。受付の再表示、通信失敗後の再試行、権限、読み上げを確認する。実行した検証と結果、実行できなかった検証、残る判断を分けて報告する。

依頼例は本資料の提案。組織のAI利用条件に従い、原記録・仕様・実際の動作で生成結果を照合する。[6]

電話が減った理由まで確かめる

確認の電話が減っても、利用者が諦めたためかもしれません。件数と、仕事が進んだ事実を一緒に見ます。

見る対象	今回の指標の例	数字と一緒に確かめること
依頼者の仕事	状態と連絡予定を 説明できた人の割合	必要な報告へ進めたか。 未定を誤解していないか
業務全体の負担	依頼100件当たりの確認電話、 重複受付、差し戻しの件数	電話の目的や依頼の種類。 別部署の作業が増えたか
開発と運用	実装・レビュー・手戻りの時間、 障害対応と保守の時間	変更の規模や難しさ、 AIの費用、担当者の偏り

導入前後は対象と期間を揃え、件数だけでなく分母を残します。小さな試行の結果は仮説を見直す材料とし、繁忙期や担当者の変化も確認します。

指標は実践提案。行動観察・問い合わせ・業務記録の併用は[10]、AI作業時間の測定上の論点は[5]を参照。

試す範囲と戻す方法を先に決める

利用者の状況を全て把握してから作り始める必要はありません。影響を限定し、分かったことから目的や仕様を直せるようにします。

試す前

対象となる部署・依頼・期間と、確認する担当者を決める。今のやり方も記録し、新しい表示の意味が伝わるかを試作で確かめる。

続けるかを決める材料

本人が必要な報告へ進めるか、受付担当の入力が増えないかを確認する。情報漏えい・受付消失・重複手配など、直ちに止める条件は業務責任者と開発者で決める。

止めた後

誰が新機能を止め、誰が利用者へ案内するかを決める。保存済みの依頼は保持し、旧画面でも扱えるかを確認する。コードを戻すだけでデータの変更が戻るとは限らない。

小さく試すことと、利用者が困ったときに仕事を続けられることを両立する。

試行の運用例は本資料の提案。小さな変更・段階的な公開と評価の考え方を参照。[11]

開発者が背景を知る機会を組み込む

現場の情報が届かなければ、開発者は確認する前提を見落とすことがあります。利用者と技術の間を行き来できる体制をつくります。

一緒に決めること	主に持ち寄る材料	残すもの
利用者・業務責任者 何を優先するか	実際の行動、期限、 受け渡し、業務上の制約	望む状態と 残る不便の理由
開発・品質の担当者 どう成り立たせるか	データ、権限、失敗時の 動作、既存機能への影響	設計の理由と 検証した結果
運用を担う人 どう使い続けるか	状態の更新、不具合、 問合せ、保守にかかる時間	担当者、見直す日、 中止と復旧の手順

一人に全てを任せる必要はありません。開発者が観察や問い合わせの振り返りに参加し、疑問を現場へ返せる時間と相談先を用意します。

役割分担は実践提案。開発者が利用者の調査に直接関わる考え方を参照。[3]

作業の自動化率から採用人数は決まらない

採用の要否は、必要な仕事の量・質・継続性と、今の体制で担える範囲によって変わります。コードを書く時間が減った後の仕事も見ます。

実際に起きていること	次に確かめること	体制の選択肢
実装待ちが減り、 検証と保守も回る	休暇・退職時の代替、 新しい依頼の量	既存体制で続ける範囲を 確認し、採用時期を見直す
生成物が増え、 レビュー待ちが長い	変更の大きさ、 設計・検証を担う力	仕事の分け方の改善、 配置・採用・外部協力
何を作るかが 定まらない	利用者へ接する機会、 業務の判断者の有無	業務理解と技術をつなぐ 役割を補う

開発が容易になることで、これまで着手できなかった仕事が増える可能性もあります。一方で、必要な実装作業が減る場合もあります。自社の仕事の変化を測り、採用・配置・育成を決めます。

体制判断の提案。採用数の予測ではない。引用したAI研究から、開発職の採用が一律に不要になるとは導けない。[4][5]

判断した理由と確かめた結果を見る

備品修理の依頼記録と、小さな修正課題を渡します。完成画面に加え、何を根拠に作り、どこまで検証したかを説明してもらいます。

評価する観点	説明してもらおうこと	確認できる行動
背景の理解	電話の目的について、 何が分かり何が未確認か	記録を示し、必要な質問を 具体的に挙げられる
設計の理由	保存と通知をどう分け、 通信失敗をどう扱うか	必要なデータと 失敗時の動きを説明できる
検証と引継ぎ	AIの提案をどこで確かめ、 何がまだ試せていないか	実行結果を示し、 次の担当へ不明点を渡せる

例：60分を、記録の確認15分、修正と検証30分、説明15分に配分。職務に合わせて範囲を調整し、AI・時間・質問への回答条件を揃えます。話し方の印象だけで評価しないよう、実際の記録を残します。

選考方法の提案。妥当性を検証済みの採用テストではない。若手と経験者で期待する責任の範囲を調整する。

仕事を終えた経験を理解へつなげる

AIで課題を終えられても、後で不具合を見つけ、直せるかは別に確かめる必要があります。育成では、その経験を意図して用意します。

短期の実験で観測されたこと

52人の開発者が未習得のPythonライブラリを使った無作為化比較実験で、直後の理解度テストの平均はAI群50%、AIなし群67%でした。差は17ポイントです。作業時間の差は統計的に有意ではありませんでした。[12]

そこから言える範囲

対象は特定の学習課題と直後の理解です。長期の熟達や、全てのAI利用の影響は分かりません。AIへの質問の仕方と成績の関連も、因果効果を確定したものではありません。

現場で用意する経験

本人が動作を予測し、コードと保存記録を追い、予測との違いを説明する。小さな不具合を自分で切り分け、先輩が根拠を一緒に確かめる。AIの利用と、本人が理解を示す機会を組み合わせます。

研究はShen・Tamkin（2026）のプレプリントと著者の解説。育成方法は本資料の提案。[12]

一つの変更を公開後まで担当して学ぶ

学ぶ対象を小さくし、利用者の場面から変更後の結果まで経験できるようにします。次は4週間で進める配分例です。

時期	本人が担うこと	先輩が確かめること
1週目	利用場면을観察し、 事実・背景の仮説・問いを残す	人物評価に置き換えていないか。 別の説明も考えたか
2週目	目指す状態を確認し、 仕様と期待する結果を書く	相手の仕事と技術の条件が つながっているか
3週目	小さく実装し、 失敗時の動きを検証する	AIの提案を読み、 自分で試した根拠があるか
4週目	限定して試し、 利用結果と負担を振り返る	想定と異なる事実から 次の変更を考えられるか

先輩の観察・レビュー時間も計画に含め、本人の判断を代行しすぎない。

4週間は例示であり、習熟を保証する期間ではない。影響の小さい変更から始め、担当範囲を経験に合わせる。

なぜ作るかと何を確認めるかを残す

案件名： _____

記入者・確認日： _____

1 本人が進めたい仕事と、その後に受け取る人

2 見た行動・聞いた言葉・記録の出典と日時

3 背景の仮説と、まだ確かめていないこと

4 本人と共有した良い状態と、優先する理由

5 今回の変更範囲と、保存・権限・失敗時の条件

6 期待する結果と、実際に行う検証

7 利用後に見る結果・増やしたくない負担

8 判断・運用・見直しの担当者と日付

未確認を空白で隠さず、次に誰へ何を確認めるかを書く。9～15ページの架空例を参考に使用。

利用者理解とAIの仕事への影響

[1] GOV.UK Service Manual | 2016年公開・2017年更新

Learning about users and their needs

利用者の目的・行動・制約を調べ、開発中も確かめ続けるための実務指針。

[2] GOV.UK Service Manual | 2017年9月公開

Contextual research and observation

通常的环境・道具・資料と、支援や運用を含む文脈を観察する方法。

[3] DORA | 参照時点の公式ガイド

User-centric focus

利用者の目的、直接のフィードバック、開発者の調査参加を開発へつなぐ考え方。

[4] DORA／Jessica Baolin・Nathen Harvey | 2026年3月10日

Balancing AI tensions: Moving from AI adoption to effective SDLC use

Google開発者の2025年第3四半期の自由回答1,110件を分析。生成と確認の負担に関する定性的研究。

AIと開発するときの設計と検証

[5] METR／Joel Beckerほか | 2026年2月24日

We are Changing our Developer Productivity Experiment Design

AIの生産性効果の追試で生じた、参加者・課題の選択と時間計測の問題を報告。

[6] GitHub Docs | 参照時点の公式文書

Application card: GitHub Copilot Agents

作業範囲、文脈、受入条件の共有と、生成物・レビュー結果を検証する際の留意点。

[7] Amazon Builders' Library／Malcolm Featonby | 参照時点の公式技術解説

Making retries safe with idempotent APIs

同じ要求の再試行による副作用を防ぐ設計。要求識別子、原子的な保存、異なる意図の区別を扱う。

[8] OWASP Cheat Sheet Series | 参照時点の公式技術ガイド

Authorization Cheat Sheet

必要な権限の設計、リクエストごとの検証、サーバー側の制御と権限のテスト。

利用の評価と運用と学習

[9] W3C WAI | WCAG 2.2の解説文書

Understanding SC 4.1.3: Status Messages

画面内で変わる受付結果などを、読み上げ等の支援技術にも伝える考え方。規範本文ではなく解説。

[10] GOV.UK Service Manual | 2018年8月6日

Measuring the success of your service

指標、利用者調査、問い合わせ等を併用し、一連の仕事で使いやすさを確かめる指針。

[11] Google SRE Workbook 第16章 | 2018年刊行

Canarying Releases

変更の影響を一部へ限定し、評価して公開を進める方法。小さな変更と戻せる運用を参照。

[12] Judy Hanwen Shen・Alex Tamkin | 2026年1月28日初稿・2月1日v2

How AI Impacts Skill Formation

52人の開発者による新しいライブラリの学習実験。短期の理解度を測定し、長期の熟達は無確認。

著者による解説（2026年1月29日）

制作：リクエスト株式会社 甲畑 智康 組織行動科学®

表紙は前作のAI生成イラストを使用。実在の利用者・研究参加者ではありません。