EVM と日時実績ログを用いた個人タスク進捗モデルの提案と実装

明星大学情報学部情報学科
長研究室
22J5-114 野々村空河

2026 年 1 月 12 日

目 次

1　はじめに

長期的な学習タスクや継続的なプロジェクトにおいて，日々の進捗を正確に把握し，適切なタイミングで行動を修正することは容易ではない。多くの既存の ToDo アプリはタスクの登録や締切管理を主眼として設計されており，進捗が計画からどの程度乖離しているのか，また現状の作業ペースで締切に間に合うのかといった情報を，必ずしも十分に提供していない。特に，進捗効率や完了予測といった定量指標を，日次実績ログに基づいて継続的に再計算し提示する機能は一般的ではない。その結果，利用者が遅延に気づくのは締切直前となり，必要な行動修正が後手に回るという問題が生じやすい。

このような課題は，利用者が自身の作業状況を把握し，計画との差を評価し，行動を修正するという一連の自己調整（Self-regulation）プロセスを継続的に回すことの難しさに起因すると考えられる。自己調整は，教育工学・学習科学分野において広く知られている概念であり，現状を把握する自己モニタリング，計画との乖離を評価する自己評価，および行動を修正する自己制御からなる循環過程として説明される。しかし，これらの過程を日常的なタスク管理の中で継続的に実践することは容易ではなく，従来のタスク管理ツールでは，この循環を十分に支援する情報設計がなされていない。

一方，プロジェクト管理分野では Earned Value Management（EVM）が広く用いられており，実績データに基づいて進捗を定量的に評価し，遅延の早期検知や完了見込みの予測を行う体系が確立されている [1]。進捗効率を示す指標である SPI（Schedule Performance Index）や，現在の作業ペースを維持した場合の完了予測日を表す EAC（Estimate at Completion）は，計画と実績の乖離を直感的に把握する上で有用である。しかし，従来の EVM は，計画値が比較的固定的であり，実績の更新頻度も限定的である企業プロジェクトを前提として設計されている。個人の学習タスクのように，作業量が日々変動し，実績が日次単位で蓄積される状況では，これらの指標をそのまま適用することは難しい。個人タスクにおいては，日次で更新される実績に基づき，進捗評価や将来予測を継続的に再計算し，適切なタイミングでフィードバックを行う仕組みが不可欠である。

本研究の新規性は，Earned Value Management（EVM）を個人タスク管理へ応用するにあたり，従来のプロジェクト管理において前提とされてきた「固定的な計画値」に依存する枠組みから脱却し，日次実績ログに基づいて進捗指標を動的に再計算する点にある。従来の EVM は，事前に定義された計画値（PV）を基準とし，実績の更新頻度も限定的であるため，進捗の乖離や遅延リスクが顕在化するまでに時間を要するという課題を持つ。これに対し，本研究では，日次で蓄積される実績データを基盤として，進捗効率（SPI）および完了予測（EAC）を継続的に更新することで，遅延状態や将来リスクを早期に可視化する「動的 EVM」の枠組みを構築する。

また，進捗指標を単に可視化するだけでは，利用者がその情報を具体的な行動修正に結びつけることは難しい。進捗の遅延や将来リスクを「認知」し，それを基に行動を選択・調整できてはじめて，自己調整は機能すると考えられる。したがって，進捗評価指標は，行動変化と結びつく形で設計・提示される必要がある。

既存の個人向けタスク管理や学習支援手法には，作業量の可視化や達成状況の提示に主眼を置いたものが多く，進捗の状態評価と将来の完了見込みを同一の枠組みで定量的に扱うことは必ずしも容易ではない。本研究では，進捗評価（SPI）と将来予測（EAC）を同一の理論枠組みで一貫して扱える点に着目し，EVM の考え方を個人タスク管理向けに再構成する。

そこで本研究では，EVM の考え方を個人タスク管理向けに再構成し，日次実績ログに基づいて進捗指標を自動的に再計算する個人タスク進捗管理モデルを提案する。本モデルでは，直近の実績ペースを表す pace7d（短期的な行動傾向を反映する指標），進捗効率を示す SPI，および将来

的な完了見込みを示す EAC を用い，SPI による遅延状態の認知，EAC による将来リスクの評価，さらにそれらに基づき，日単位で実行可能な行動へ落とし込む機構として「今日のプラン」を提示することで，利用者の自己調整プロセス全体を支援することを目指す。

本研究は，進捗指標の精度そのものを評価することを主目的とするのではなく，実利用ログに基づき，これらの指標やプラン提示が，利用者の遅延認知や行動選択にどのような影響を与え，行動修正につながり得るかに着目する点に特徴がある。

本論文では，上記提案モデルを実装した進捗管理アプリを開発し，実際の利用ログを用いた多角的な分析を通じて，提案手法の有効性を検証する。

本論文の主な貢献は，以下の 3 点にまとめられる。

- **動的 EVM モデルの再構成**：日次実績ログを前提に，pace7d・SPI・EAC を連動させて継続的に再計算する進捗管理モデルを構築した点。
- **自己調整理論に基づく行動支援設計**：進捗認知（SPI），将来リスク認知（EAC），行動提示（今日のプラン）を一貫した枠組みで設計し，通知も含めて実装した点。
- **実利用ログに基づく行動変化の分析**：実際のユーザ利用データを対象に，遅延警告や EAC 超過提示後の作業量変化，および計画遵守傾向を定量的に検証した点。

これらの貢献を踏まえ，本研究では「進捗指標が行動修正を誘発し得るか」を中心的な問いとして検証を進める。

2　研究の目的

本研究の目的は，個人の長期タスクに対して「遅延の早期検知」「行動修正」「進捗理解の促進」を実現する進捗管理モデルを構築することである。本モデルは，日次実績ログに基づく定量的な進捗評価を通じて，利用者が自身の作業状況や将来リスクを早期に把握し，行動修正を行えるよう支援することを目的とする。

第一章で述べたように，従来の ToDo アプリはタスクの登録や締切管理に重点が置かれており，計画と実績の乖離や将来の締切リスクを日常的に判断するための情報が十分に提供されていない。また，従来の EVM は固定的な計画値を前提としており，作業計画が日々変動する個人の長期タスクへそのまま適用することは難しい。

本研究では，このような課題に対し，固定的な計画値に基づく進捗管理ではなく，日次実績ログから算出される動的な指標を用いることで，個人タスクに適した進捗評価と将来予測を行う。具体的には，実績ペースや必要ペースに基づく進捗効率指標（SPI）と，予測完了日（EAC）を継続的に更新することで，計画と実績の乖離を定量的に把握し，行動修正につなげることを目指す。ただし，本研究は作業時間の見積精度そのものの向上を目的とはせず，乖離が生じることを前提とした支援に主眼を置く。

以上の目的を達成するため，本研究では以下の点を具体的な研究目標として設定する。特に，自己調整（Self-Regulation）プロセスの主要な要素である自己モニタリング，自己評価，自己制御の各段階を支援するため，技術的目標と理論的目標を以下に定める。

- 個人タスクにおける進捗状態を定量化し，遅延リスクを早期に認知可能な進捗評価モデルを構築する。

- 進捗評価結果を行動判断に結びつけるためのフィードバック機構を設計し，利用者の行動修正を支援する。
- 将来の完了見込みを継続的に可視化することで，行動と進捗予測の関係を理解しやすくする。
- 自己調整（Self-Regulation）理論に基づき，自己モニタリング・自己評価・自己制御の循環を継続的に支援する進捗管理モデルの理論的基盤を確立する。

本論文の評価設計は，目的と RQ，貢献の対応を以下のように整理している。目的 1（遅延の早期検知・行動修正）は RQ1・RQ2，貢献 1・2 に，目的 2（進捗理解の促進）は RQ3 および貢献 3 に紐づく。各章では，この対応関係に沿って指標と手法を選定する。

上記の目的を踏まえ，検証可能な研究課題（Research Questions）を以下に定義する。

- **RQ1（遅延認知と行動変化）**：SPI が警告閾値を下回る状態を提示した後，翌週の平均作業時間または pace7d は有意に増加するか（閾値は 4.4.6 を参照）。
- **RQ2（将来リスク認知と行動変化）**：EAC が締切を超過すると提示された直後，翌週の締切逸脱リスク（EAC 超過日数，必要ペースの超過度合い）は改善するか。
- **RQ3（行動提示と実行）**：「今日のプラン」で提示された推奨作業量 todayMinutes と，実際の当日投入時間との乖離はどの程度か。乖離が小さいほど進捗回復（SPI の改善）に寄与するか。

以降の実験章では，これらの RQ に対応する指標と分析方法を明示し，日次実績ログを用いて検証する。なお，RQ1・RQ2 で扱う「警告」は，SPI 閾値や EAC 超過を検知したタイミングでシステムが UI と通知に表示可能な状態として生成したイベントを指し，利用者が実際に閲覧したかどうかはログ上で区別していない。閲覧の有無を含む介入強度の差分は，今後追加計測すべき制約として位置づける。

本研究は，EVM 指標を日次実績ログと組み合わせて動的に運用し，自己調整理論に基づくフィードバックを提供する点に特徴を有する。この理論的立場および目的設定は，第 4 章で述べるシステム設計と，第 5 章で行う評価・分析の前提となる。

3　関連研究

近年，個人向けのタスク管理サービスは数多く提供されており，Google ToDo，TickTick，Notion，Trello などが広く利用されている。これらのサービスは，タスクの登録，締切設定，通知といった基本機能を通じて，ユーザーの作業整理や忘却防止を支援している。

一方で，多くのサービスは，進捗の定量的な把握や，見積もりと実績の比較に基づく支援までは十分に踏み込んでいない。たとえば TickTick はポモドーロタイマーによる作業ログ記録を提供しているが，見積時間 E と実績時間 A の体系的な比較や，進捗効率指数（SPI）や予測完了日（EAC）といった定量指標を用いたフィードバックには対応していない。また，Notion や Trello のようなデータベース型サービスでは柔軟なタスク管理が可能であるものの，進捗の自動評価やペース分析といった機能はユーザー自身の設計に委ねられている。

表 1 に，代表的なタスク管理サービスと進捗管理機能の有無を比較した結果を示す。なお，本研究システムについても未実装機能が存在するため，比較が特定のシステムに過度に有利とならないよう留意している。

表 1: 主要タスク管理サービスと進捗管理機能の比較

機能項目	Google ToDo	TickTick	Notion	Trello	本研究
タスク登録・締切設定	◯	◯	◯	◯	◯
見積時間（E）の設定	×	△	△	×	◯
実績時間（A）の記録	×	◯	△	×	◯
SPI/EAC の表示	×	×	×	×	◯
pace7d に基づく評価	×	×	×	×	◯
作業提案通知	×	△	×	×	△
コラボレーション	×	◯	◯	◯	×
スマホ UI 対応	◯	◯	◯	◯	△（PC 特化）

表 1 は，本研究システムを含む代表サービスの機能有無を「実績時間の収集」「進捗指標の提示」「行動提案」の 3 軸で整理したものであり，未実装機能がある本研究システムが不当に有利にならないよう，評価軸を限定し注記を付した。SPI 閾値については，PMI や NASA のガイドライン [7] にみられる SPI < 0.8 を重大遅延とみなす運用を参考にし，本研究でも同じ値を採用する。

3.1 学習者の自己調整と進捗把握の困難性

学習科学の分野では，学習者が自身の進捗や作業量を正確に把握することの難しさが繰り返し指摘されている。野上ら [2] は，学習者が自らの学習量を過大または過小に評価する傾向を報告しており，主観的な自己評価のみに基づく学習管理には限界があることを示している。

一方で，学習時間や作業ログといった客観的な行動データを可視化することは，学習行動の調整（self-regulation）を促進する要因となることが知られている。これらの知見は，行動データに基づく定量的なフィードバックが，学習や作業の自己管理において重要な役割を果たすことを示唆している。

3.2 自己調整学習モデルと行動フィードバック

自己調整（Self-regulation）は，学習者や作業者が目標達成に向けて自身の行動を調整する過程として，教育工学および心理学の分野で広く研究されてきた。SRL 研究の第一人者である Zimmerman [4] は, 自己調整を目標設定から行動修正までを循環的に辿るプロセスとして体系化した。DiBenedetto and Zimmerman [5] は, この SRL プロセスにおけるセルフモニタリング（自己観察）と自己評価（Self-evaluation）の実施頻度が, 学習者の学業成績と正の相関を持つことを実証的に示しており, 客観的な進捗データに基づく継続的なフィードバックが目標達成能力の向上に寄与することを示している。

これらの研究では，行動の記録（モニタリング），進捗や差異の評価，それに基づく行動修正が循環的に行われることが重要であるとされている。このような枠組みは，進捗管理システムを設計する上での理論的背景として位置づけられる。

3.3 作業時間見積もりの誤差に関する研究

作業時間の見積もりに関しては，実プロジェクトにおいて，不確実性による遅延の可能性を考慮して活動期間を見積もることが極めて重要である。Morita and Suwa [6] は，スケジュールを安定させることを目的とした活動期間の見積もり手法を提案し，従来の PERT や Critical Chain/Buffer Management（CC/BM）といった手法と比較することで，見積もり方法の違いがスケジュール安定性に大きな影響を与えることを示した。

これらの研究から，見積もりプロセス自体が困難であり，不確実性によって見積値と実績値の間に乖離が生じることは避けられない。このため，初期の見積もり誤差を前提とした動的な進捗管理の必要性が示唆される。

3.4 EVM の個人タスクへの応用と課題

Earned Value Management（EVM）は，プロジェクト管理において計画と実績の差異を定量的に評価する手法として広く用いられており，SPI（Schedule Performance Index）や EAC（Estimate at Completion）といった指標が知られている。

しかし，従来の EVM は企業プロジェクトを主な対象として設計されており，個人の学習タスクのように日々の実績が大きく変動する文脈では，そのまま適用することが難しい。具体的には，

- 計画値（PV）が柔軟な個人計画に適合しづらい
- 実績ログが日単位で更新されるため，指標の再計算頻度が高い
- 完了見込みの変化が短期間で大きく変動しうる

といった課題が挙げられる。

実務においては，PMI や NASA，米国国防総省（DoD）などの EVM ガイドライン [7] において，SPI ≥ 1.0 を計画通り，$0.8 \leq \text{SPI} < 1.0$ を軽度の遅延，SPI < 0.8 を重大な遅延とする基準が用いられる例が多い。これらの基準は，進捗リスクを定量的に把握する指標として広く参照されている。

プロジェクト管理分野には，WBS やガントチャート，バーンダウンチャートなど，計画立案や進捗可視化に有用な手法が多数存在する。しかし，これらの手法は，進捗の状態評価と将来の完了見込みを同一の指標体系で扱うことを主目的としておらず，日次実績ログに基づいて継続的に更新し，行動修正へ直接結びつける枠組みは限定的である。

本研究では，進捗評価（SPI）と将来予測（EAC）を一体的に扱える点に着目し，EVM の考え方を個人タスク管理へ応用した。EVM は本来，企業プロジェクトを対象とした手法であるが，その構造は日次実績ログにも適用可能であり，進捗認知と行動修正を結びつける基盤として適していると考えられる。

このように，EVM は進捗評価と将来予測を同一の指標体系で一体的に扱える点において，他のプロジェクト管理手法にはない特徴を有している。一方で，その設計前提は固定的な計画値と比較的低頻度の評価に基づくものであり，個人タスク管理への適用には再構成が必要である。

本研究では，EVM 指標そのものの精度向上を目的とするのではなく，日次実績ログを前提として指標を継続的に再計算することで，進捗状態の認知と将来リスクの把握を可能とし，それらを行動修正へと結びつける動的なフィードバック機構として EVM の考え方を再構成している点に特徴がある。

3.5 Self-regulation 理論と進捗管理システム

教育工学・心理学の分野では，自己調整（Self-regulation）は，以下の 3 段階からなる循環的プロセスとして整理されることが多い。

- Self-monitoring：自身の行動や進捗を把握する段階
- Self-evaluation：計画や目標との差を評価する段階
- Self-control / adjustment：評価結果に基づき行動を修正する段階

表 2 は，これらの各段階と，進捗管理システムにおいて想定される機能との対応関係を整理したものである。

表 2: Self-regulation 段階と進捗管理システム機能の対応

Self-regulation 段階	内容	想定される機能例
Self-monitoring	現状把握	作業ログ記録
Self-evaluation	評価・将来予測	進捗指標・予測表示
Self-control / adjustment	行動修正	作業計画の見直し

3.6 関連研究の整理

以上の関連研究から，既存の個人向けタスク管理サービスは，タスクの登録や締切管理，作業履歴の可視化を主眼としており，進捗の定量評価や将来予測に基づく行動修正を一体的に支援する枠組みは限定的であることが分かる。また，EVM に基づく進捗管理手法は，進捗評価と将来予測を同一の指標体系で扱える点に強みを有する一方で，主に企業プロジェクトを対象として設計されており，固定的な計画値を前提とする点で，個人の長期タスクへの適用には課題が残る。さらに，自己調整理論は学習や作業の自己管理において有効性が示されているものの，日次実績ログに基づく進捗指標と結び付けて進捗の認知から行動修正までを一体的に実装・評価した研究は多くない。

本研究は，これらの先行研究の限界を踏まえ，EVM の考え方を日次実績ログに基づく動的な進捗管理モデルとして再構成し，自己調整プロセスにおける進捗認知と行動修正を一体的に支援する点に新規性を有する。

4 システムの実装

本章では, 本研究で開発した進捗マネジメントアプリの構成と実装について述べる。まず開発環境を示し, 次にシステム全体の構造, データモデル, 主要機能,UI 設計について順に説明する。

4.1 開発環境

本研究のアプリケーションは，Web ブラウザ上で動作するシングルページアプリケーション（SPA）として実装した。利用者が日次で実績を入力し，その結果に応じて進捗指標を即座に再計算・提示する必要があることから，リアルタイム性と保守性を重視した構成を採用している。

主な使用技術は以下のとおりである。

- フロントエンド：React（Vite）
- 言語：JavaScript / JSX
- バックエンド：Firebase（Authentication, Firestore, Cloud Functions）
- 通知：Firebase Cloud Messaging（FCM）
- 実行環境：Node.js
- バージョン管理：Git / GitHub

Firebase は，認証，データベース，バックエンド処理（Cloud Functions），通知（FCM）を統合的に提供しており，日次実績ログの更新を契機とした進捗指標の自動再計算や，利用者入力に即応したフィードバック処理を実装しやすい点に利点がある。このため，本研究で提案する「実績入力 → 評価 → 行動支援」という進捗管理モデルの実装基盤として適していると判断し，採用した。

4.2 システム構成

本システムは,React を用いたフロントエンド,Firestore を中心とするデータベース層, そして Google Cloud Functions を用いたバックエンド処理の 3 層構成で動作する。図 1 に全体構成を示す。

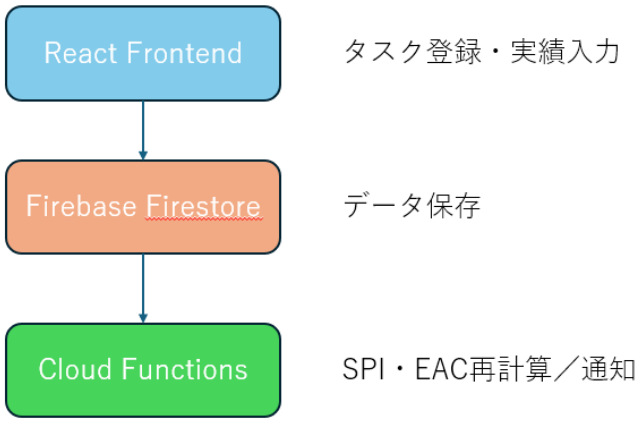

図 1: 本研究システムの全体構成

ユーザー操作はフロントエンドで行われ, データは Firestore に保存される。Firestore の更新をトリガとして Cloud Functions が進捗指標の計算や通知スケジューリングを自動実行することで, ユーザーの入力に応じて常に最新状態の進捗データが維持される。

フロントエンドでは, タスク登録や実績入力, 分析画面での可視化といったユーザーとのインタラクションが中心となる。タスク一覧画面には,EVM 指標（SPI・EAC・残作業量など）がリアルタイムに反映され, 各タスクの遅延リスクは SPI や EAC に基づいて「良好」「注意」「遅延」に色分けされて表示される。

一方,Firestore ではタスクデータ・日次実績ログ・必要ペース（requiredPace）・SPI・EAC といった EVM 指標を保持する。日次ログは actualLogs として日付キーで管理され, 後述する pace7d の計算に利用される。

Cloud Functions では Firestore の変更を監視し, 以下の主要処理を自動実行する。

- updateStats.js：日次実績ログが更新されるたびに,pace7d, 残作業量, 必要ペース,SPI,EAC を再計算し Firestore に書き戻す。ログ編集にも対応しており, 過去の実績が修正された場合でも全期間の SPI および EAC を整合的に再計算する。
- scheduleMorningSummary.js：毎朝, ユーザー設定の時刻（`morningPlanTime`）と一致した場合にのみ実行し, 推奨タスクを計算したうえで dailyPlans ドキュメントへ保存してから FCM を送信する（メッセージは簡潔に留め, 詳細はアプリ内で確認させる）。
- scheduleProgressReminder.js：夜間リマインド時刻（`progressReminderTime`）と一致し, かつ当日の実績が 1 分も記録されていない場合にだけ送信する。未入力ユーザーへのみに限定することで過剰通知を抑制している。

この 3 層構成により, ユーザーが実績入力を行うだけで EVM 指標の自動更新・通知処理・可視化が連鎖的に行われ, 進捗管理の循環（記録 → 評価 → フィードバック）が自動的に機能するアプリケーション構造が実現されている。

4.3 データモデル

本研究のアプリケーションは，タスク情報・日次実績ログ・EVM 指標を総合的に管理するために，Firestore を用いた階層型データモデルを採用している。図 2 に，本アプリで用いる主要データ構造の概要を示す。本モデルは，ユーザーが実績を入力するたびに Cloud Functions が自動的に再計算を行い，最新の進捗指標が常に Firestore 上に反映されることを前提として設計している。また，入力経路の追跡のために sessions サブコレクション（source/trigger を含む）を保持し，日次メトリクス（通知送信回数や日次 EVM 指標）を users/*uid*/*metrics*/date に蓄積している。

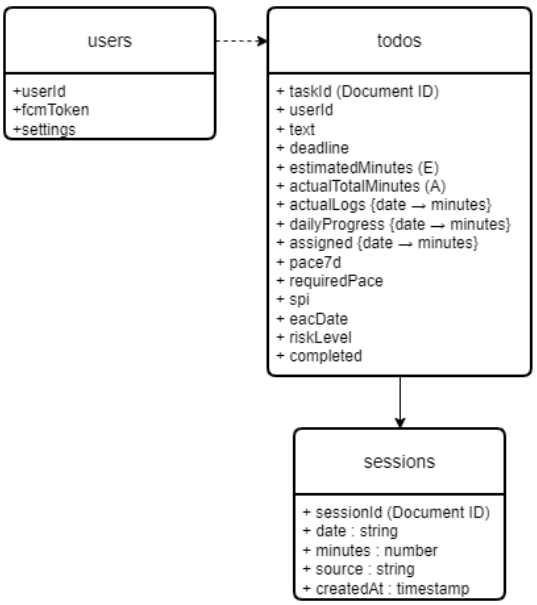

図 2: Firestore のデータモデル（タスク・実績ログ）

本研究のアプリでは，進捗管理に必要な情報を以下のように保持する。

- **todos コレクション（タスク）**
  - text：タスク名
  - estimatedMinutes（E）：見積時間
  - actualTotalMinutes（A）：累積実績
  - actualLogs：日次実績ログ（YYYY-MM-DD: 分）
  - pace7d：直近 7 日間の平均実績ペース
  - requiredPaceAdj：期日までに完了するための必要ペース
  - spi：進捗効率指数（SPI）
  - eacDate：予測完了日（EAC）
  - deadline：締切日時
  - completed：完了フラグ
  - createdAt：タスク登録日時
  - lastProgressAt：最後に実績が記録された日時
  - userId：タスク所有ユーザ ID
- **sessions サブコレクション**
  - date：実績入力日
  - minutes：入力された作業時間
  - source：入力経路（手動 / 通知から）
  - createdAt：記録時刻
- **users コレクション**
  - fcmToken：通知送信に使用するトークン
  - settings：通知 ON/OFF などの設定
  - loginCount：利用回数（継続性の指標）

特に actualLogs と pace7d は本研究特有のデータ構造であり，EVM 指標を日次ベースで再計算するための基盤となっている。

4.4 主要機能の実装

本節では，本研究で開発した進捗マネジメントアプリにおける主要機能について述べる。以下では，タスク登録，実績入力，進捗指標の更新，実績ログの編集，およびデータのエクスポートといった処理を中心に，各機能がどのように連携して実装されているかを説明する。

表 3: todos コレクションの主なフィールド（改訂版）

フィールド名	概要
`text`	タスク内容（例：IT パスポート 100 時間勉強する）。
`deadline`	タスク全体の締切日時 D。
`estimatedMinutes`	見積総学習時間 E［分］。進捗率・残作業量算出の基礎となる。
`actualLogs`	日付ごとの実績時間 $A(t)$ を保存するマップ。pace7d，EAC，SPI の再計算に使用。
`actualTotalMinutes`	累積実績時間 $\sum_t A(t)$［分］。
`dailyProgress`	日別の累積進捗率（その日までの EV ／ 総 EV）。
`actualProgress`	現時点の累積進捗率（EV ／ 総 EV）。
`idealProgress`	線形計画に基づく理想進捗率。計画との差分（遅れ判定）に利用。
`pace7d`	直近 7 日間の平均学習ペース［分／日］。SPI・EAC の基礎指標。
`requiredPace`	遅れを考慮しない基本の必要ペース（残作業量 ÷ 残日数）。
`requiredPaceAdj`	遅れ状況（idealProgress と actualProgress の差）を考慮し，遅れ量を残日数で均等に取り戻す前提で算出される調整後必要ペース［分／日］。
`spi`	SPI（進捗効率指数：pace7d ／ requiredPaceAdj）。
`spi7d`	直近 7 日間の SPI。pace7d と requiredPaceAdj から算出される。初期 3 日間は作業日数で割って pace を計算するウォームアップを行う。
`eacDate`	現在のペースを続けた場合の予測完了日（EAC）。
`riskLevel`	遅延リスクのカテゴリ（“ok”, “warn”, “danger” など）。
`completed`	タスク完了フラグ。
`createdAt`	タスク登録日時。
`lastProgressAt`	最後に実績が記録された日時。
`userId`	タスク所有ユーザ ID。

4.4.1 タスク登録（TodoInput.jsx）

ユーザーがタスク名，締切日時，見積時間（E）を登録すると，Firestore の `todos` コレクションに新規ドキュメントが作成される。登録された見積時間 E は，後続の進捗指標計算処理において参照され，残作業量や必要ペース，SPI，EAC などの各指標の算出に利用される。また，タスクはラベルや作成日時，ユーザー ID とともに保存され，後述する分析機能に利用される。

4.4.2 実績入力（ProgressEntry.jsx）

ユーザーが 1 日の作業時間を入力すると，以下の処理が自動で行われる。

1. `actualLogs[日付]` への実績加算
2. `actualTotalMinutes`（累積実績 A）の更新
3. `sessions` サブコレクションへの入力ログ保存

記録された日次実績は，pace7d（直近 7 日平均）の算出に利用され，SPI や EAC などの進捗指標の再計算に反映される。また，実績は後から修正可能であり，過去ログの変更にも対応している。

4.4.3 進捗指標計算（updateStats.js）

Firestore の実績ログ（`actualLogs`）が追加・更新・削除されるたびに，Cloud Functions により進捗指標の再計算処理が実行される。本研究では，日次実績ログにもとづいてタスクの進捗状態を継続的に更新するため，指標計算ロジックを updateStats.js に集約した。

本処理は，ユーザーの実績入力や過去ログの修正に応じて即時に起動し，各タスクについて残作業量，直近の作業ペース，進捗効率指数（SPI），および完了予測日（EAC）などの主要指標を再

算出する。算出結果は Firestore の該当フィールドに書き戻され，フロントエンドでの可視化，リスクレベル判定，および「今日のプラン」生成などの後続処理から参照可能な状態に保たれる。

また，本研究では過去ログの編集を許容する設計を採用しているため，指標は差分更新ではなく，日次実績ログに基づく再計算として整合性を担保する。これにより，入力の誤り訂正や後日入力が生じた場合でも，全期間にわたって一貫した指標値が得られる。

各指標の算出規則（例外値の扱いを含む）および数式定義は，次節の「進捗指標の数式定義（実装準拠の動的 EVM）」に示す。さらに，updateStats.js 内の計算手順や条件分岐の詳細は，付録 A においてアルゴリズムとして整理する。

4.4.4 進捗指標の数式定義（実装準拠の動的 EVM)

本節では，前節で述べた `updateStats.js` により再計算される進捗指標について，本研究における正式な数式定義および例外条件を，実装仕様と対応付けた形で示す。本研究の進捗管理モデルは，日次実績ログを基盤として進捗評価および将来予測を継続的に更新する「動的 EVM（Earned Value Management）」の枠組みに基づいている点に特徴がある。

以下では，タスク i に対し，日付 d 時点での各指標を定義する。

(1) 累積実績および残作業量　タスク i の総見積時間を E_i，d 日目までの累積実績作業時間を $A_i(d)$ とする。このとき，残作業量 $R_i(d)$ は次式で定義される。

$$R_i(d) = \max\{0,\, E_i - A_i(d)\}. \tag{1}$$

(2) 締切までの残日数　締切日を D_i とし，d 日目における残日数 $L_i(d)$ は，締切との差を日数に換算した上で切り上げ，最小値を 1 日とする。

$$L_i(d) = \max\left\{1,\, \left\lceil \frac{D_i - d}{1\ \text{day}} \right\rceil \right\}. \tag{2}$$

(3) 直近平均ペース（pace7d）　直近 7 日間における日次実績作業時間を $a_i(k)$ とすると，d 日目の直近平均ペース $\mathrm{pace7d}_i(d)$ は次式で定義される。ただし，実装では初期利用段階における不安定な評価を避けるため，実績が存在する日数が 3 日未満の場合には，分母を実働日数とするウォームアップ補正を行う。

$$\mathrm{pace7d}_i(d) = \begin{cases} \dfrac{\sum_{k=d-6}^{d} a_i(k)}{7} & (\text{実績日数} \geq 3), \\[2ex] \dfrac{\sum_{k=d-6}^{d} a_i(k)}{\text{実績日数}} & (\text{実績日数} < 3). \end{cases} \tag{3}$$

(4) 必要ペース　締切までに残作業量を消化するために必要な 1 日あたりの作業量（基本必要ペース）は，次式で定義される。

$$\mathrm{requiredPace}_i(d) = \begin{cases} \dfrac{R_i(d)}{L_i(d)} & (L_i(d) > 0), \\[2ex] 0 & (\text{otherwise}). \end{cases} \tag{4}$$

(5) 調整後必要ペース　実装では，進捗警告閾値 θ_{relax} （ユーザー設定値，未設定時は 0.9）を下回る場合に限り，必要ペースを緩和する。緩和係数 ρ_i （ユーザー設定値，未設定時は 0.9）を用い，調整後必要ペースを次式で定義する。

$$\mathrm{requiredPaceAdj}_i(d) = \begin{cases} \mathrm{requiredPace}_i(d) \cdot \rho_i & (\mathrm{SPI}_i(d) < \theta_{\mathrm{relax}}), \\ \mathrm{requiredPace}_i(d) & (\text{otherwise}). \end{cases} \tag{5}$$

(6) 進捗効率指数（SPI）　進捗効率指数 SPI は，直近平均ペースと基本必要ペースの比として定義される。ただし，基本必要ペースが正の値を取らない場合には，以下の例外処理を行う。

$$\mathrm{SPI}_i(d) = \begin{cases} \dfrac{\mathrm{pace7d}_i(d)}{\mathrm{requiredPace}_i(d)} & (\mathrm{requiredPace}_i(d) > 0), \\ 1 & (R_i(d) = 0), \\ \text{undefined} & (\text{otherwise}). \end{cases} \tag{6}$$

なお，リスクレベル判定に用いる固定閾値（$\theta_{\mathrm{risk}} = 0.8$）と例外的判定規則は 4.4.6 に示す。

(7) 予測完了日（EAC）　計画開始日 $\mathrm{plannedStart}_i$ が未設定，または $d \geq \mathrm{plannedStart}_i$ を満たす場合を `startReached` とする。EAC は，$\mathrm{pace7d}_i(d) > 0$ かつ `startReached` が真の場合にのみ算出され，残作業量を現在のペースで消化するのに必要な日数を切り上げて求める。

$$\mathrm{EAC}_i(d) = d + \left\lceil \frac{R_i(d)}{\mathrm{pace7d}_i(d)} \right\rceil. \tag{7}$$

以上のように，本研究では，日次実績ログに基づいて進捗指標を動的に再計算することで，個人タスクにおける遅延状態や将来リスクを継続的に可視化する進捗管理モデルを構築している。なお，本節で定義した進捗指標の数式および例外条件は，本文中では本節に集約して用いる。

4.4.5　EAC 予測日の算出と履歴的可視化

本研究では，予測完了日（EAC）を，単一の固定値として保持する指標ではなく，日次実績ログに基づいて動的に再計算される将来予測指標として位置づけている。

updateStats.js は，実績ログ（`actualLogs`）が追加・更新・削除されるたびに，その時点の直近作業ペースを用いて EAC を再算出し，Firestore 上の `eacDate` フィールドとして最新値を保持する。これにより，利用者は，現在の作業ペースを維持した場合にタスクがいつ完了すると見込まれるかを，常に最新の状態で把握できる。

一方で，本研究では，EAC の過去値を時系列データとして永続的に保存する設計は採用していない。これは，EAC が実績の蓄積状況に応じて変動する予測値であり，過去ログの編集や修正によって遡及的に変化し得る性質を持つためである。過去の予測値を固定的に保存すると，実績ログとの不整合が生じる可能性がある。

そこで本研究では，EAC の履歴的な推移を，値の保存によって管理するのではなく，日次実績ログを基盤として必要に応じて再計算・再構成する方式を採用した。具体的には，フロントエンドの可視化および分析処理において，各時点の実績ログから EAC を再算出することで，予測値の時系列的変化を表現している。

この「再計算可能性を重視した設計」により，過去ログの編集が行われた場合であっても，全期間にわたって一貫した前提条件のもとで EAC の推移を再評価することが可能となる。また，EAC

の変動そのものを利用者の進捗認知や将来リスク認知の変化として捉えることで，自己調整理論における**自己評価（Self-evaluation）**を支援する役割も担っている。

以上のように，本研究における EAC は，固定的な予測値ではなく，日次実績ログに基づいて更新され続ける動的な将来予測指標として設計されており，進捗管理モデル全体における中長期的な行動調整の判断材料として機能する。

4.4.6 リスクレベル判定（SPI 基準）

本システムでは，進捗効率指数（SPI）を進捗状態を定量的に把握するための主要指標として用い，その値に基づいてタスクのリスクレベルを段階的に分類する。SPI は Cloud Functions 内の判定ロジックで参照され，以下の 3 段階の状態として整理される。

- $\mathrm{SPI} \geq 1.0$：進捗良好
- $0.8 \leq \mathrm{SPI} < 1.0$：注意
- $\mathrm{SPI} < 0.8$：遅延

ここで用いられる閾値 0.8 は，進捗遅延のリスクを判定するための固定閾値（$\theta_{\mathrm{risk}} = 0.8$）であり，前節で定義した進捗指標の数式とは独立に設定されている。本研究では，$\mathrm{SPI} < 0.8$ の状態を「計画に対して著しい進捗遅れが生じている状態」と解釈する。

また，本システムでは，SPI に基づく判定に加えて，予測完了日（EAC）が締切日時を超過すると判定された場合，あるいは締切日時を過ぎてもタスクが完了していない場合には，SPI の値に関わらず遅延状態として扱う。これは，短期的な進捗効率が一時的に高い場合であっても，将来的な締切遵守が困難である状況を適切にリスクとして認識させることを目的とした例外的判定である。

これらの判定結果は，`riskLevel` として Firestore に保持され，後述するユーザーインタフェース層において，色分け表示や注意喚起の形で利用者に提示される。

4.4.7 当日の作業計画におけるタスク選定アルゴリズム

本節では,「今日のプラン」機能において，当日に取り組むタスクをどのように選定しているかについて概要を述べる。本機能は，利用者が進捗状況に基づいて行動を計画・修正する，自己調整理論における自己制御（Self-Control）プロセスを直接的に支援することを目的として導入されたものである。

「今日のプラン」は，利用者がその日に着手すべきタスクを把握しやすくするために，進捗指標および締切情報をもとに，当日の作業候補を最大 3 件提示する機能である。最大 3 件とするのは，タスク切替コストを抑えて行動負荷を下げることを意図した実装上のヒューリスティックである。

タスク選定処理は，フロントエンドおよびバックエンドの双方から利用可能な共通モジュールとして実装されており，実行環境に依存しない一貫した判定結果が得られる構成としている。

アルゴリズムでは，まず未完了であり締切日時が設定されているタスクを候補として抽出する。この際，遅延状態にあるタスクのみを対象とするのではなく，締切が近づいている進捗良好なタスクも候補に含めることで，締切直前の未着手を防ぐことを意図している。

候補タスクは，進捗状況，締切までの残期間，および残作業量といった情報を統合した優先度スコアにもとづいて評価され，そのスコア順に整列される。この優先度スコアは，遅延度合いと締切

リスクを同時に考慮できるよう設計されており，本研究におけるタスク選定の中核的な判断基準となっている。その結果にもとづき，上位のタスクが当日の作業計画として提示される。

なお，候補集合の定義には，開始予定日や見積値の妥当性など，実装上の追加条件が含まれるが，これらの詳細な条件や優先度スコアの数式的定義については，付録 B において詳述する。

4.4.8 推奨作業時間（todayMinutes）の算出

選定された最大 3 件のタスクに対し，本節では，当日に割り当てる推奨作業時間（`todayMinutes`）をどのように算出しているかを述べる。本システムでは，単に必要ペースを提示するのではなく，利用者の進捗状況と当日の作業可能時間を踏まえ，**実行可能性と行動修正の両立**を目的とした作業時間の提示を行う。

推奨作業時間の算出は,「最低限の達成ライン」「進捗回復を考慮したライン」「余剰キャパシティの活用」という三段階の考え方にもとづいて行われる。日次キャパシティの既定値は 120 分であり，利用者設定（dailyCap）で上書き可能とした。

(1) 日次キャパシティの確定　当日に利用可能な作業時間の上限を日次キャパシティ C［分］とする。基本的にはユーザーが設定した日次キャパシティ（`dailyCap`）を用いるが，未設定，0 以下，または不正値の場合には，安全策として既定値 120 分を用いる。

当日計画の算出および再計算はいずれも，この日次キャパシティを上限として実行される。

(2) 第 1 パス：最低ライン（required）の割当て　各タスク i について，まず「締切までに間に合わせるために最低限必要な作業量」を最低ラインとして割り当てる。

残作業量 R_i は，見積時間 E_i から累積実績時間 A_i を差し引いた量として定義される。必要ペースは，進捗遅れを考慮した調整後必要ペース（`requiredPaceAdj`）が存在する場合はそれを，存在しない場合は基本の必要ペース（`requiredPace`）を用いる。

この段階では，遅延状態にあるタスクが締切に間に合うために**最低限必要な作業量を確保する**ことを目的とする。

(3) 第 2 パス：回復ライン（recover）の割当て　次に，各タスクについて，進捗遅れの改善を目的とした回復ラインを考慮する。

本実装では，回復ラインは，進捗効率指数（SPI）を 1.0（計画通りの進捗状態）に回復させることを目標として定義している。この目標に到達するために必要な作業量を回復目標量とし，第 1 パスで割り当てた量との差分が存在する場合には，日次キャパシティの範囲内で追加の割当てを行う。

これにより，遅延状態にあるタスクについては，単に締切を遵守するだけでなく，**進捗状態そのものの回復を見据えた**作業量が提示される。

(4) 第 3 パス：余剰キャパシティの配分　第 2 パス終了後も作業キャパシティが残っている場合には，締切が近い順にタスクを並べ替え，前倒し的に作業を進めるための追加割当てを行う。

この余剰配分は，1 タスクあたり最大 30 分を上限とする制約のもとで行われ，当日の余剰時間を有効活用しつつ，将来的な作業負荷の軽減を図るための補助的な割当てとして位置づけられる。

(5) フォールバック処理　上記の三段階の割当てを経ても，当日の作業計画が空となる場合に限り，フォールバック処理を行う。

この場合，候補タスクの範囲内で最大 3 件を対象とし，残作業量と締切までの残日数に基づいて日割りで作業量を割り当てる。本処理は,「今日のプランが空になる」状態を回避するための最小限の補助的処理として位置づけられる。

(6) 小括　以上のように，推奨作業時間の算出は，最低限の達成ライン，進捗回復ライン，および余剰時間の活用という三段階の構成に基づいて行われる。特に，最低ライン（required）と回復ライン（recover）を段階的に分離して割り当てる設計は，進捗遅れの克服と行動修正を同時に支援する点において，本研究における行動支援アルゴリズムの中心的な役割を担っている。これにより，利用者の当日の作業可能時間と進捗状態を反映した，実行可能かつ行動修正を促す作業計画の提示を可能としている。

4.4.9　当日計画の再計算処理

本節では,「今日のプラン」に対して再計算が行われる条件と，その際の処理内容，および本研究における位置づけについて述べる。

本システムでは，利用者の作業状況の変化を即座に計画へ反映することを強制するのではなく，利用者自身の判断にもとづいて当日計画を見直せる設計を採用している。当日計画の再計算処理は，**タスク選定および時間割当アルゴリズムが持つ動的な設計思想を維持したまま**，前節までに述べたアルゴリズムを，利用者が必要と判断した時点の最新状態にもとづいて再適用する処理として位置づけられる。

(1) 再計算のトリガ　当日計画の再計算は，利用者が明示的に「今日のプランを更新」操作を行った場合に実行される。

作業実績の入力は，進捗指標の更新や利用者自身の進捗認知を促す役割を担うが，それ自体が当日計画の自動的な再計算を引き起こすことはない。再計算は，利用者が計画の修正を必要と判断した任意のタイミングで実行される設計となっている。

(2) 再計算時の処理内容　再計算時には，前節までに示したタスク選定アルゴリズムおよび推奨作業時間算出アルゴリズムを，当日の最新の進捗指標にもとづいて再実行する。割当てに用いる日次キャパシティは初期計算と同じ設定値（未設定時は 120 分）をそのまま上限として使用し，当日に既に投入した実績時間を上限から控除する処理は実装していない。

具体的には,

- 未完了タスクを対象として候補集合を再生成する
- 優先度評価にもとづき，当日に取り組む最大 3 件のタスクを再選定する
- 日次キャパシティ設定値を上限として，三段階の割当方式にもとづき推奨作業時間（`todayMinutes`）を再算出する

この結果，再計算後に提示される当日計画は，利用者がその時点で再検討することを選択した「当日の作業計画」として更新される。

(3) 再計算処理の位置づけ　本研究における当日計画の再計算処理は，計画の最適化や頻繁な変更を目的とするものではない。むしろ，作業実績の入力によって進捗状況を可視化し，その結果を踏まえて利用者自身が計画の更新を選択できる状態を提供することで，主体的な行動調整を支援することを目的としている。

すなわち，本処理は，実績入力による**自己モニタリング**，進捗評価にもとづく**自己評価**，および更新された計画にもとづく**自己制御**という，自己調整理論における循環過程を，日々の作業計画の中で具体化するための**動的な行動支援機構**として位置づけられる。

(4) 小括　以上より，本システムの当日計画再計算処理は，利用者の判断にもとづいてタスク選定および時間割当アルゴリズムを再適用する機構であり，作業実績と行動計画を柔軟に接続する役割を担っている。この構造により，計画提示と実績入力を往復する自己調整の循環を支援し，日々の行動修正を促す進捗管理モデルを実現している。

4.4.10　通知機能（Cloud Functions）

本システムでは，利用者の自己調整的な学習・作業習慣を支援するために，Cloud Functions および Firebase Cloud Messaging（FCM）を用いた通知機能を実装している。本機能は，単なる時刻ベースのリマインダではなく，進捗管理モデルの運用を補助し，利用者が計画確認や実績入力といった行動を適切なタイミングで行えるよう促すことを目的としている。

本研究では，通知によって行動を直接指示・強制するのではなく，利用者自身の判断による行動修正を支援する補助的手段として通知機能を位置づけている。そのため，通知の種類と内容は必要最小限に留めている。

(1) 朝のプラン確認通知　毎朝所定の時刻に Cloud Functions がユーザデータを読み込み，当日の進捗状況や残作業量をもとに「今日のプラン」を算出する。その後，FCM を通じてユーザ端末へ通知を送信し，アプリ内の「今日のプラン」画面を確認するよう促す。

本実装では，

> 「今日のプランを確認してみましょう。」

といった簡潔なメッセージのみを送信する設計としている。これは，通知内で具体的なタスクや作業量を提示するのではなく，詳細な判断はアプリ画面上で行わせることで，利用者の主体的な計画確認を促すことを意図したものである。

(2) 夜の実績入力リマインド　当日の作業実績が未入力の場合には，夜間に実績入力を促す通知を送信する。本研究で用いる進捗指標（SPI，EAC，pace7d など）は，日次の実績ログを前提として算出されるため，実績入力の継続は進捗管理モデル全体の信頼性を保つうえで重要である。

この通知は，作業量そのものを増やすことを目的とするものではなく,「振り返り」と「記録」という自己モニタリング行動を習慣化することを主な目的としている。

(3) 通知生成と計画再計算との関係　作業実績の入力や過去ログの編集が行われた場合には，Cloud Functions により進捗指標が再計算される。ただし，本システムでは，当日計画の再計算（4.4.7）や指標更新が行われた場合であっても，即時に通知を送信することは行わない。

これは，通知頻度の過剰化による負担を避け，利用者が必要と感じたタイミングで自発的に計画更新を行う余地を残すためである。計画や指標の変化は，次回の定期通知（朝または夜）を通じて間接的に反映される。

(4) 本機能の位置づけ　通知機能は，利用者が「計画を確認する」「実績を記録する」という行動の起点を提供するものであり，自己調整理論における**自己モニタリング**および**自己制御**を補助する役割を担う。

すなわち，本研究における通知は，進捗管理モデルの中核であるタスク選定・作業時間算出アルゴリズムを代替するものではなく，それらの機能が有効に機能するための**行動誘発的インタフェース**として位置づけられる。

4.4.11　ログ編集機能（LogEditorModal）

本システムでは，ユーザーが過去に入力した作業実績を後から修正できるログ編集機能を実装している。実績の修正が行われた場合には，Cloud Functions が自動的に呼び出され，日次実績ログを基準として pace7d，requiredPaceAdj，SPI，EAC，riskLevel といった進捗指標が再計算される。

本研究では，実績入力の誤りや後日入力といった実利用環境における不確実性を考慮し，過去のログを遡って修正した場合であっても，進捗指標の整合性が維持される設計を採用した。具体的には，指標計算を単純な差分更新ではなく，全期間の日次データにもとづいて再実行することで，履歴全体に対する一貫した進捗評価を可能としている。

この設計により，利用者は記録の正確性を保ちながら継続的に利用でき，また研究用途においても，信頼性の高い時系列データを取得できる。

4.4.12　データエクスポート機能

本研究では，提案システムの評価および分析を行うため，記録された作業実績ログや進捗指標を CSV 形式でエクスポートできる機能を実装した。

エクスポート対象には，日次の実績時間，累積実績，SPI，EAC，requiredPaceAdj などの指標が含まれており，これらを外部の分析環境で利用することが可能である。本機能により，通知への反応傾向や進捗指標の推移，見積誤差の変化といった，提案モデルの有効性を検証するための定量的な分析が可能となる。

このように，本システムは日常的な進捗管理を支援するアプリケーションであると同時に，研究データの収集および評価を目的とした実験基盤としても利用可能な構成となっている。

4.5　UI 画面の構成

本節では，ユーザーが実際に操作する UI 画面の構成について述べる。4.4.10 節で説明した各種処理（EVM 指標の計算，リスク判定，通知処理など）はバックエンドで自動的に行われ，本節ではそれらの結果がどのように画面上で提示され，ユーザーの自己調整行動を支援しているかに焦点を当てる。

自己調整（Self-regulation）との対応関係は以下のとおりである。

- 自己モニタリング：日次ログ入力（ホーム・今日のプラン・ログ編集）と夜リマインド
- 自己評価：タスク一覧・分析画面での SPI/EAC・リスク表示，EAC 時系列グラフ
- 自己制御：今日のプラン（最大 3 件，キャパシティ上限で割当），朝のプラン通知による確認導線

本アプリの主要な画面は，カレンダー画面，タスク一覧画面，分析画面，ログ編集画面，設定画面の 5 つである。

本節では，主要な UI を図 3 〜 図 9 に示し，各画面の役割と研究上の意義を述べる。

4.5.1 カレンダー画面

カレンダー画面（図 3）では，月単位でタスクの締切日や実績入力の有無を俯瞰できる。各日付には締切を持つタスクが表示され，締切が集中している期間や，全く作業していない期間が一目で確認できる。

この画面は，ユーザーが長期的な作業スケジュールを把握し，「どの時期に負荷が偏っているか」「どのあたりから着手すべきか」といった判断を行うための手がかりを提供する役割を持つ。

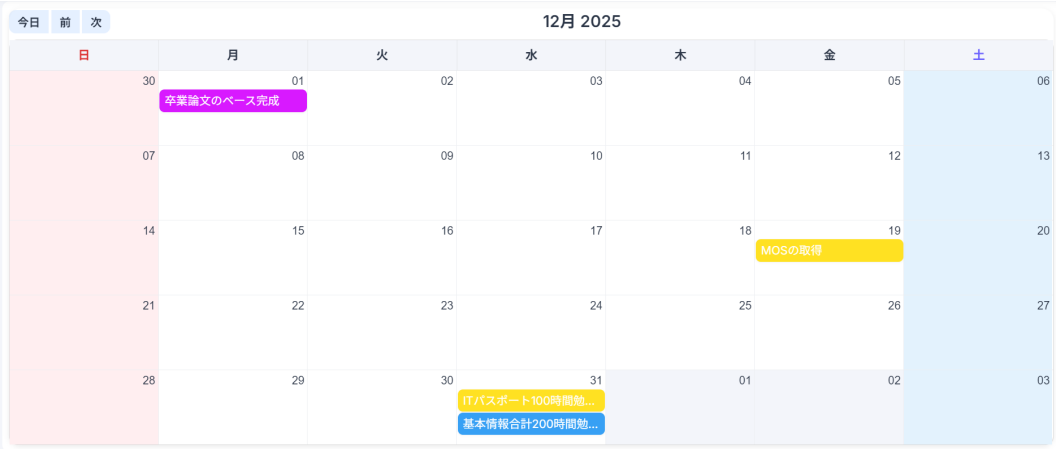

図 3: カレンダー画面（月間のタスク進捗と締切の俯瞰）

4.5.2 タスク一覧画面

タスク一覧画面（図 4）は，本アプリの中心的な画面であり，各タスクの概要と進捗状況が一覧で表示される。各カードには，タスク名，締切，見積時間，累積実績，残作業量に加えて，最新の EVM 指標（pace7d，必要ペース，SPI，EAC）が要約された形で示される。

特に，SPI に基づくリスクレベルは，カードの背景色やバッジとして視覚的に表現される。SPI が 1.0 以上のタスクは緑色（良好），0.8 以上 1.0 未満は黄色（注意），0.8 未満は赤色（遅延）として表示されるため，ユーザーは「どのタスクが危険域に入りつつあるか」を瞬時に把握できる。これにより，着手順序の決定や当日の作業計画に反映しやすくなる。

図 4: タスク一覧画面（SPI, EAC, 残量, 必要ペースなどのリアルタイム可視化）

4.5.3 分析画面

分析画面（図 5）では，EVM 指標を用いた時系列の可視化が行われる。主な表示内容は次のとおりである。

- 日別の作業時間と累積作業量
- 直近 7 日間および 30 日間の平均ペース
- タスク別の SPI 推移（折れ線グラフ）
- 理想的な進捗曲線と実績の比較

これらのグラフにより，ユーザーは「いつからペースが落ち始めたのか」「行動の変化が SPI や EAC にどのように影響したか」を視覚的に理解できる。特に SPI の折れ線グラフは，注意状態や遅延状態に入ったタイミングを振り返る材料となり，自身の学習・作業習慣をメタ認知するきっかけを与える。

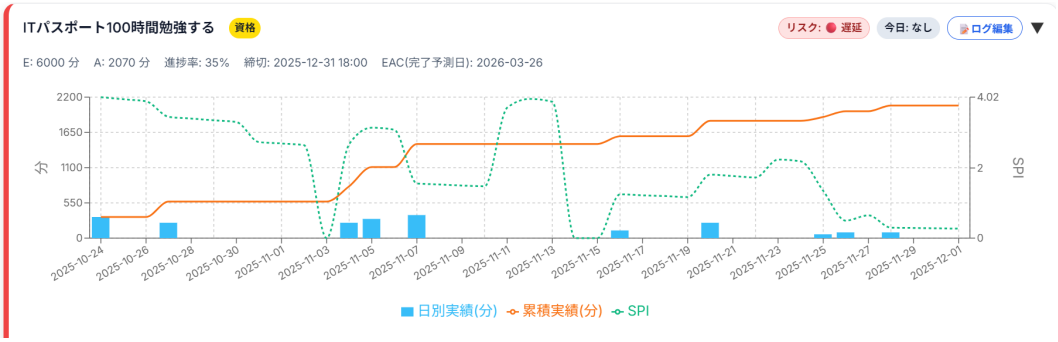

図 5: 分析画面（作業時間推移，直近平均，SPI 推移）

さらに本研究では，タスクごとに日次で再計算される EAC（予測完了日）の履歴を折れ線グラフとして可視化している。このグラフでは，縦軸を 2 つ用い，左軸に残作業量，右軸に予測完了

日（EAC）を配置することで,「残作業量の増減」と「予測完了日の遅れ・回復」を同時に把握できるようにした。

これにより，ユーザーは，

- どの時点で EAC が締切を超過し始めたか
- 作業量の増減が EAC にどのように影響したか
- 行動改善（作業ペース向上）が EAC をどの程度回復させたか

を視覚的に確認できる。

図 6 に，本研究で導入した EAC（予測完了日）の日次推移を可視化した折れ線グラフを示す。残作業量（左軸）と EAC 予測日（右軸）を同時に表示することで，実績ペースの変化が締切遅延にどの程度影響しているかを直感的に確認できる。

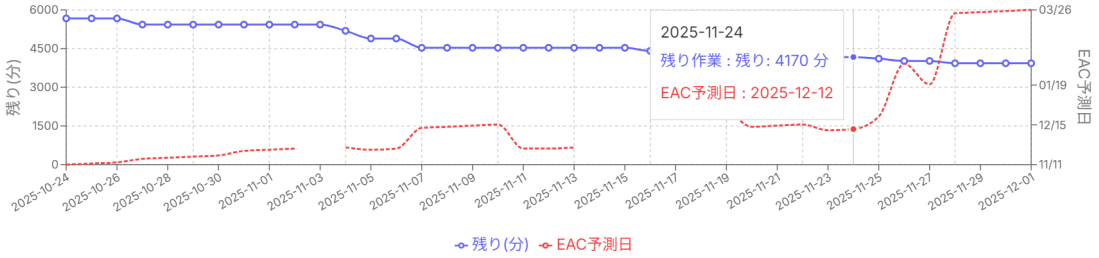

図 6: EAC 予測日の時系列グラフ（残作業量との比較）

一般的なタスク管理では締切遅延の兆候を把握しづらいが,EAC の時系列変化を見ることで「どの行動が遅れの原因となったか」「改善行動がどの程度効いたか」を直接確認でき，これは自己調整学習におけるメタ認知的フィードバックとして重要な役割を果たす。

4.5.4 ログ編集画面

ログ編集画面（図 7）では，日付ごとに記録された実績時間を一覧し，誤入力や記録漏れを修正できる。ユーザーはカレンダーや一覧から日付を選択し，当日の作業時間を増減させることができる。

この画面は，実績データの欠損や誤差を補正し，より現実に近い進捗記録を維持するためのものである。ユーザーにとっては,「ちゃんと記録できていない」という心理的なストレスを軽減し，日次記録の継続を支える役割も持つ。

4.5.5 設定画面

設定画面では，図 8 に示す画面で通知時刻や通知のオン・オフを設定でき，タスクラベルの管理，アプリのバージョン情報などを確認・変更できる。通知時刻を生活リズムに合わせて調整できるようにすることで，ユーザーは無理のないタイミングでリマインドを受け取ることができる。

また，ログイン回数やバージョン情報（コミット ID）を表示することで，利用継続の度合いやシステム更新履歴を簡単に把握できるようにしている。これらは研究上は補助的な情報であるが，ユーザーの利用意欲や再現性の確保に寄与する。

図 7: ログ編集画面（入力忘れや誤入力の修正に対応）

以上のように，本アプリの UI は，4.4 節で述べた EVM 指標計算やリスク判定の結果を，ユーザーが理解しやすい形で提示することを目的として設計されている。各画面はそれぞれ異なる時間スケール（1 日，1 週間，1 か月）や視点（タスク単位，全体の傾向）からフィードバックを与えることで，ユーザーの自己調整を多面的に支援している。

本章のまとめと次章への接続: UI とバックエンドの構造が，自己モニタリング・自己評価・自己制御の循環を支える基盤であることを示した。次章では，この構造を前提にした実験設計と指標算出方法を述べる。

- アプリの通知をユーザーの生活リズムに適応
- ラベル管理によりカテゴリ別学習の分析が可能
- バージョン情報により再現性と信頼性を担保
- ログイン回数の可視化は継続利用の動機づけになる

4.5.6 今日のプラン画面

今日のプラン画面では，4.4 節で述べたタスク選定アルゴリズムに基づいて，当日に取り組むべき最大 3 件のタスクが提示される。各タスクカードには

- 残作業量 R
- 必要ペース（requiredPaceAdj）

図 8: 通知設定画面

- 当日割り当てられた推奨作業量（todayMinutes）

が表示され，ユーザーは優先すべきタスクと作業量を直感的に把握できる。

本画面には「今日のプランを更新」ボタンが配置されており，ユーザーが任意のタイミングで当日のプランを再計算できる。再計算時のアルゴリズムは 4.4.9 節で述べたとおり，日次キャパシティ設定値（未設定の場合は 120 分）を上限として初期計算と同じ手順で割当てを再実行する。当日にすでに入力された実績時間をキャパシティから差し引く処理は行わず，最新の進捗指標とタスク情報を用いてその時点の推奨作業量を提示する。

さらに，本画面では当日の計画値（Planned）と実績値（Actual）が棒グラフによって可視化される。これによりユーザーは，その日の計画達成度を視覚的に確認でき，日中の自己調整行動を支援する役割を果たす。

図 9: 今日のプラン画面（タスク一覧と割当作業量の提示）

5　実験方法

本章では，第 6 章で行う評価・分析に先立ち，本研究における実験方法および分析の前提条件を定義する。本研究は，開発した進捗管理アプリの実利用ログを用いた介入を伴わない観察研究として実施され，利用者が日常的な学習・作業の中でアプリを利用した結果として自然に蓄積されたログデータを分析対象とする。そのため，本章で扱うデータは，実利用環境における進捗認知および行動選択の実態を反映したものとなる。

本研究では，同一の実利用ログを用い，進捗評価指標および行動支援機構が利用者の行動にどのように関与しているかを複数の観点から検証することを目的とする。

5.1　研究課題（RQ）の設定

前節で述べた目的を具体化するため，本研究では以下の三つの研究課題（RQ）を設定する。

- **RQ1：**SPI に基づく遅延状態の提示は，利用者の作業行動にどのような変化をもたらすか。
- **RQ2：**EAC によって締切超過が示唆された際，利用者の作業行動に変化が生じるか。
- **RQ3：**SPI および EAC に基づいて生成される「今日のプラン」の提示は，利用者の当日のタスク選択および作業時間配分にどのような影響を与えるか。

これらの研究課題は，実利用環境において提示される進捗情報や行動支援と，利用者の行動選択との関係を観察的に分析することを目的としている。

5.2　データセットと観測期間

本研究では，開発した進捗管理アプリの実利用により自動的に記録されたログデータを用いて分析を行う。本データセットは，アプリを日常的な学習・作業管理の目的で利用した複数の利用者による実利用ログから構成されている。

分析対象となる主なデータは，日次単位で記録された作業実績ログであり，各タスクに対して，日付ごとの作業時間が保存されている。これらのログは，進捗指標（SPI，EAC など）の算出および更新に利用され，実験期間中を通じて継続的に蓄積された。

観測期間は，2025 年 4 月 8 日から 2026 年 1 月 1 日までの約 9 か月間である。当該期間において，利用者数は 7 名，各利用者により複数のタスクが登録され，日次実績ログは合計 229 レコードが記録された。

本研究で用いるログデータには，個人を特定可能な情報は含まれておらず，すべて匿名化されたユーザ ID のみを用いて管理されている。そのため，利用者のプライバシーに配慮した形で実利用環境における行動データの分析が可能となっている。

5.3　実験手順

本研究における実験手順は，利用者による日常的なアプリ利用の流れに沿って構成されている。以下に，データが生成・記録され, 分析対象として整理されるまでの手順を示す。

まず，利用者はアプリ上で，学習や作業に対応するタスクを登録し，各タスクに対して締切日や想定作業量を設定する。登録されたタスク情報は，タスク管理データとして保存される。

次に，利用者は作業を開始する前に，過去の実績ログに基づいて算出された SPI や EAC などの進捗情報，およびそれらを踏まえて生成された「今日のプラン」を確認する。これらの情報は，前日までに蓄積された実績データを反映し，利用者が当日の作業方針を検討する際の参考情報として提示される。

その後，利用者は実際に学習・作業を行い，作業終了後に，当日に実施した作業内容について，タスクごとの作業時間を日次実績として入力する。この実績入力を契機として，当該タスクに関する進捗指標（SPI，EAC など）が再計算され，翌日以降のプラン生成に反映される。

本研究では，この実績入力を一つの更新単位として，進捗指標が変化した日を分析上の「入力イベント」として扱う。また，EAC の算出結果として予測完了日が締切日を超過すると判定された場合，当該日を「遅延予測イベント」と定義する。

分析にあたっては，これらのイベント発生日を基準として，同一タスク内におけるイベント前後の一定期間の実績作業時間を集計した。具体的には，イベント発生日の直前および直後それぞれについて，過去 7 日間の作業時間の平均値を算出し，前後比較を行うことで，行動変化を評価した。

以上の一連の過程を通じて，タスク情報，日次実績ログ，進捗指標，およびプラン提示に関する情報が継続的に記録される。本研究では，このように実利用の中で自然に生成されたログデータを用いて，研究課題に対応する分析を行う。

5.4 分析対象データの抽出条件

本研究では，実利用ログの中から，研究課題に対応した行動変化を分析するため，以下の条件に基づいて分析対象データを抽出した。

分析対象とするのは，観測期間中に記録されたタスクに関するログである。各タスクについて，日次実績ログが記録されている日を分析単位とし，作業時間が入力されていない日は，行動量の変化を評価できないため分析対象から除外した。

また，進捗指標（SPI，EAC）および「今日のプラン」提示と作業行動との関係を検証するため，指標の更新やプラン提示が行われた日を基準として，同一タスク内における前後の行動変化を分析対象とした。異なるタスク間での直接的な比較は行わず，各タスクを独立した分析単位として扱っている。

EAC に基づく分析では，将来予測指標 EAC の算出結果として，予測完了日がタスクの締切日を超過すると判定された日を「遅延予測イベント」と定義し，当該イベントを分析対象として抽出した。

さらに，補足的な分析として，遅延予測イベントが発生した日において，利用者のログインが確認された場合と確認されなかった場合とで，行動変化の傾向を比較した。ログインの有無は，進捗指標や行動支援情報が閲覧可能な状態であったかを近似的に表す条件として用いている。

分析 3 では,「今日のプラン」に基づく行動選択との整合性を検証するため，当日プランに含まれるタスクを planned task，含まれないタスクを unplanned task と定義した。また，当日に日次実績ログに作業時間が記録されたタスクを，当日実行されたタスクとして扱い，当日プランと実行行動との対応関係を分析対象とした。

以上の条件に基づき抽出されたログデータを用いて，以降の評価・分析を行う。

5.5 評価指標と統計的検定方針

本研究では，進捗情報および行動支援の提示が利用者の作業行動にどのように関与しているかを検証するため，実利用ログから算出可能な指標を評価指標として用いる。

評価指標 分析 1 では，進捗評価指標である SPI に着目し，日次実績の入力を一つの更新単位として，実績入力前後における SPI の変化量 $\Delta\mathrm{SPI} = \mathrm{SPI}_{\mathrm{after}} - \mathrm{SPI}_{\mathrm{before}}$ を主要な評価指標とする。特に，当日朝時点で $\mathrm{SPI}_{\mathrm{before}} < 1$ を満たす遅延状態からの進捗回復に着目し，ΔSPI の分布および符号を分析する。

分析 2 では，将来予測指標である EAC に基づき，予測完了日が締切日を超過すると判定された日を遅延予測イベントとして定義する。各イベントについて，イベント発生日の前後一定期間（7 日間）における 1 日あたり平均作業時間を算出し，前後差を評価指標として用いる。

分析 3 では，SPI および EAC に基づいて生成される「今日のプラン」と，当日の実行行動との整合性を評価する。具体的には，当日プランに含まれるタスク（planned task）と含まれないタスク（unplanned task）に分け，それぞれについて当日の実行有無を判定することで，タスク実行率を評価指標とする。また，planned task に対しては，計画作業時間に対する実績作業時間の比率（実績作業時間／計画作業時間）を算出し，時間遵守率として，当日の時間配分の整合性を評価する。

統計的検定方針 統計的検定においては，同一タスク内における実績入力前後の比較を基本とし，対応のあるデータとして扱う。実利用ログに基づくデータは，分布の正規性を仮定できない場合が多いため，原則として分布を仮定しないノンパラメトリック検定を用いる方針とした。

具体的には，分析 1 における ΔSPI の符号および分布の偏りについて，Wilcoxon の符号付順位検定を用いて検証する。一方，分析 2 および分析 3 では，サンプル数が限られることや，欠損を含む場合があることから，仮説検定の適用が困難となるため，記述統計に基づく傾向の整理を主とする。

各分析における具体的な検定手法および結果については，第 6 章において示す。

6　結果・考察

6.1　分析 1：SPI に基づく遅延状態からの進捗回復効果

本分析では，SPI に基づいて判定される遅延状態からの進捗変化に着目し，実利用ログに記録された実績入力前後の SPI の変化を分析する。

まず，利用者による日次実績の入力を一つの入力イベントとみなし，当該イベント直前に算出されていた SPI を $\mathrm{SPI}_{\mathrm{before}}$，実績入力後に更新された SPI を $\mathrm{SPI}_{\mathrm{after}}$ と定義する。また，これらの差分 $\Delta\mathrm{SPI} = \mathrm{SPI}_{\mathrm{after}} - \mathrm{SPI}_{\mathrm{before}}$ を，進捗状態の変化量として用いる。

さらに，直近 7 日間の実績に基づいて算出される SPI が安定して評価可能な状態を週間評価フェーズ *weekly* と定義する。具体的には，過去 7 日以内に 3 日以上の実績入力が存在し，それらの実績に基づいて SPI が算出されている場合を *weekly* として扱う。なお，週間評価フェーズに該当しない入力イベントは，短期評価フェーズ（*short*）として扱う。本分析では，当該フェーズ内で完結する入力イベント（weekly→weekly）を主な分析対象とする。

図 10 にそれぞれの概念図を示す。

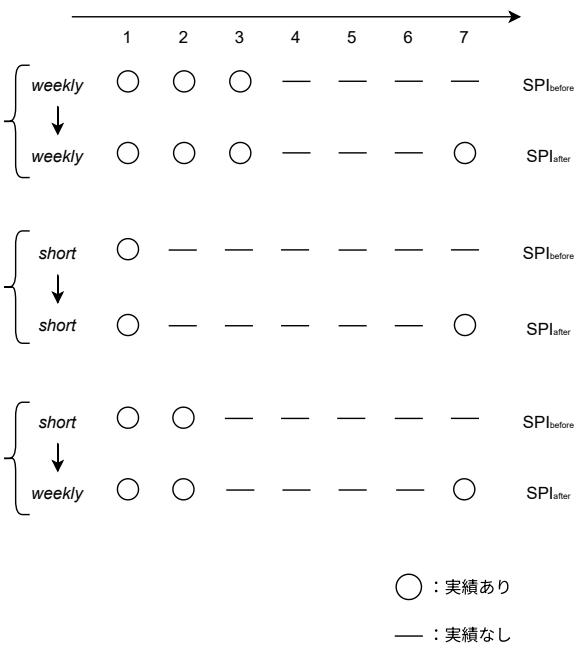

図 10: SPI_before および SPI_after の算出範囲と評価モード遷移の概念図

特に，当日朝時点で $\mathrm{SPI}_{\mathrm{before}} < 1$ を満たす遅延状態のケースに注目し，実績入力を通じて進捗回復が生じているかを観察的に検証する。

6.1.1　分析の目的と仮説

本分析は，第 5 章で設定した研究課題 RQ1「SPI に基づく遅延状態の提示は，利用者の作業行動にどのような変化をもたらすか。」を検証することを目的とする。

特に，週間評価フェーズにおける遅延状態（$\mathrm{SPI}_{\mathrm{before}} < 1$）では，実績入力を契機として ΔSPI が正の値をとるケースが多いと期待される。

6.1.2　データ抽出と前処理

観測期間中に記録された日次実績ログ 229 件を対象とし，前章で定義した分析対象条件に基づいてデータの抽出および前処理を行った。

まず，$\Delta\mathrm{SPI} \neq 0$ となった日を入力イベントとして抽出した結果，178 件の入力イベントが得られた。

次に，各入力イベントについて，実績入力前後の評価フェーズに基づきモード遷移（*weekly* / *short*）を付与した。その結果，weekly→weekly が 109 件，short→short が 53 件，short→weekly が 16 件となった。

本分析では，実績入力前後の評価フェーズがともに *weekly* である入力イベント（weekly→weekly）を主な分析対象とする。データ抽出の各段階における件数の推移を表 4 に示す。

表 4: 分析 1 におけるデータ抽出の段階別件数

抽出段階	件数
生データ（日次ログ）	229
欠損・締切起因の除外後	186
$\Delta\mathrm{SPI} \neq 0$ の入力イベント	178
うち weekly→weekly	109
（$\mathrm{SPI}_{\mathrm{before}} < 1$）	38
うち short→short	53
うち short→weekly	16

6.1.3　主分析：weekly→weekly における SPI 改善効果

主分析では，weekly→weekly の入力イベントのうち，当日朝時点で $\mathrm{SPI}_{\mathrm{before}} < 1$ を満たす遅延状態のケース 38 件を対象とした。

これらのケースにおける ΔSPI の記述統計量を表 5 に示す。ΔSPI の中央値は 0.081，平均は 0.104 であり，すべてのケースにおいて $\Delta\mathrm{SPI} > 0$ が観測された。

すなわち，weekly 評価フェーズにおける遅延状態では，実績入力後に算出される SPI が，入力前の SPI を上回るケースが一貫して確認された。

表 5: weekly→weekly かつ $\mathrm{SPI}_{\mathrm{before}} < 1$ に限定した ΔSPI（主分析対象）

指標	値
件数 n	38
平均 ΔSPI	0.104
中央値 ΔSPI	0.081
$\mathrm{pos\_rate}(\Delta\mathrm{SPI} > 0)$	1.00

6.1.4　統計検定による改善方向の検証

主分析で対象とした weekly→weekly の入力イベントにおいて，ΔSPI が正の値をとる傾向が偶然によるものではないかを検証するため，統計的検定を行った。

検定対象は，weekly→weekly の入力イベント 109 件とし，各入力イベントにおける ΔSPI を用いた。同一タスク内での実績入力前後の比較であり，また ΔSPI の分布に正規性を仮定できないことから，対応のあるノンパラメトリック検定として Wilcoxon の符号付順位検定を採用した。検定は，$\Delta\mathrm{SPI} > 0$ を対立仮説とする片側検定として実施した。

その結果，weekly→weekly では ΔSPI が正の方向に有意に偏っており，$p = 9.56 \times 10^{-15}$ が得られた。また，効果量は $r = 0.73$ と，比較的大きな値を示した。

参考として，評価フェーズのモード遷移ごとに ΔSPI の集計および Wilcoxon 検定を行った結果を表 6 に示す。

表 6: ΔSPI のモード別集計と Wilcoxon 検定結果

モード遷移	件数	$\mathrm{pos\_rate}(\Delta\mathrm{SPI} > 0)$	$\Delta\tilde{\mathrm{SPI}}$	$\tilde{\mathrm{SPI}}_{\mathrm{before}}$	$\tilde{\mathrm{SPI}}_{\mathrm{after}}$	p_{Wilcoxon} (r)
weekly→weekly	109	0.95	0.324	1.554	1.885	9.56×10^{-15} (0.73)
short→short	53	0.91	1.500	0.000	2.000	3.43×10^{-9} (0.80)
short→weekly	16	0.13	-0.981	1.780	0.735	1.00 (-0.79)

weekly→weekly では，$\Delta\mathrm{SPI} > 0$ となる割合が 0.95 と高く，中央値および平均ともに正の値を示した。一方で，short→short では正方向の割合は高いものの ΔSPI のばらつきが大きく，short→weekly では ΔSPI が負方向に偏る傾向が確認された。

これらの結果から，評価尺度の一貫性が保たれる weekly→weekly の入力イベントを主分析対象とすることが妥当であると判断した。

6.1.5　補足分析：モード別・条件別の挙動

主分析では，weekly→weekly に限定した遅延状態において，ΔSPI が一貫して正となる傾向を確認した。本節では，この結果の解釈を補足するため，評価モードの違いおよびログイン条件に応じた ΔSPI の挙動を整理する。

表 7 に，評価モード別およびログイン条件別の ΔSPI の記述統計を示す。weekly→weekly では，$\Delta\mathrm{SPI} > 0$ となる割合が高く，中央値および平均値はいずれも正の値を示した。また，イベント数も最も多く，週間スケールでの評価において，進捗改善が比較的安定して生じていることが確認できる。

一方，short→short では，中央値および平均値が weekly→weekly よりも大きく，短期間で大きな改善が生じるケースが含まれている。ただし，ΔSPI のばらつきも大きく，短期評価フェーズにおける指標変動の不安定さが示唆される。また，short→weekly のように評価フェーズの切替を伴う入力イベントでは，ΔSPI が負方向に偏る傾向が観測された。これは，評価期間の分母や算出条件の切替に起因する指標上の変動を含んでいる可能性を示している。

さらに，ログイン有りの weekly→weekly 入力イベントでは，全体と比較して ΔSPI の中央値および平均値が大きい傾向を示した。これは，進捗指標や行動支援情報を閲覧可能な状態にあった入力イベントにおいて，より大きな進捗改善が観測されていたことを示している。

ただし，本研究では，利用者が実際に $\mathrm{SPI}_{\mathrm{before}}$ や関連情報を閲覧したかどうかを直接的に確認するログは取得していない。そのため，本節で示した結果は，SPI の提示と行動改善との因果関

係を示すものではなく，主分析結果の解釈範囲を明確化するための補足的な整理として位置づけられる。

表 7: 評価モードおよびログイン条件別の ΔSPI の記述統計（補足分析）

評価モード	条件	件数 n	中央値 ΔSPI	平均 ΔSPI	pos_rate(ΔSPI > 0)
weekly→weekly	全イベント	109	0.324	0.730	0.95
weekly→weekly	ログイン有り	118	0.442	1.22	–
short→short	全イベント	53	1.500	1.89	0.91
short→weekly	全イベント	16	-0.981	-0.52	0.13

6.1.6 考察

本分析では，SPI に基づいて遅延状態と判定された入力イベントに着目し，実績入力前後における進捗状態の変化を観察的に検証した。その結果，週間評価フェーズ（weekly）において $\mathrm{SPI}_{\mathrm{before}} < 1$ を満たす遅延状態のケースでは，すべての入力イベントで ΔSPI > 0 が観測され，実績入力後に SPI が改善する傾向が一貫して確認された。

また，weekly→weekly の入力イベント全体に対する Wilcoxon 符号付順位検定においても，ΔSPI が正の方向に有意に偏っていることが示され，週間評価フェーズにおける SPI が，実績入力を通じて改善方向に更新されやすい指標であることが示唆された。これは，直近 7 日間の実績に基づく評価という週間スケールの集約が，日次の作業量変動を過度に反映せず，進捗状態を比較的安定して捉えているためであると考えられる。

一方で，短期評価フェーズ（short）や評価フェーズの切替を伴う入力イベントでは，ΔSPI のばらつきが大きく，特に short→weekly のようなケースでは，ΔSPI が負方向に偏る傾向が観測された。これらの変動には，評価期間の分母や算出条件の切替に起因する指標上の変化が含まれている可能性があり，ΔSPI の解釈には注意が必要である。この点からも，評価尺度の一貫性が保たれる weekly→weekly の入力イベントを主分析対象とした判断は妥当であったといえる。

さらに，補足分析において，ログイン有りの入力イベントでは，ΔSPI の中央値および平均値が全体よりも大きい傾向が確認された。これは，進捗指標や行動支援情報を閲覧可能な状態にあった場合に，より大きな進捗改善が生じていた可能性を示唆する結果である。ただし，本研究では，利用者が実際に $\mathrm{SPI}_{\mathrm{before}}$ や関連情報を閲覧したかどうかを直接的に確認するログは取得していない。そのため，SPI の提示が行動改善を直接的に引き起こしたと因果的に結論づけることはできない。

以上より，本分析は，SPI に基づく遅延状態において，実績入力を通じて進捗状態が改善方向に更新される傾向が実利用環境下で観測されることを示した。これは，SPI が遅延状態からの進捗回復を捉える評価指標として一定の妥当性を有していることを示唆する結果であり，次節以降では，将来予測指標や行動支援機構との関係について，さらに検討を行う。

6.2 分析 2：EAC に基づく将来予測と行動変化

6.2.1 分析の目的

本分析は，第 5 章で設定した研究課題 RQ2「EAC によって締切超過が示唆された際，利用者の作業行動に変化が生じるか」を検証することを目的とする。

将来予測指標である EAC は，現在までの実績ペースをもとに将来的な完了見通しを提示する指標であり，SPI のような短期的な進捗評価とは異なる性質を持つ。そのため，EAC に基づく遅延

予測の提示は，作業量の即時的な増加だけでなく，作業計画の見直しや行動方針の再検討といった形で利用者の行動に影響を与える可能性がある。

本分析では，EAC に基づく遅延予測イベントを基準として，イベント前後における作業量の変化を比較することで，将来予測情報が利用者の作業行動にどのような変化をもたらしているかを実利用ログに基づき観察的に検証する。

6.2.2　遅延予測イベントの概要

実験方法章で定義した条件に基づき抽出した結果，観測期間中に確認された遅延予測イベントは 27 件であった。

これらのイベントは，同一タスク内において EAC の算出結果が締切超過を示唆した更新時点を基準としており，将来的な完了見通しに関する再評価が生じたタイミングを表している。

分析 1 で扱った SPI に基づく入力イベントが，日次実績入力に伴う短期的な進捗状態の変化を対象としていたのに対し，本分析で扱う遅延予測イベントは，現在までの作業ペースを踏まえた中長期的な完了予測の変化に着目する点に特徴がある。

以下では，遅延予測イベント前後における作業量の変化を整理し，EAC による将来予測情報と利用者の作業行動との関係を検討する。

6.2.3　遅延予測イベント前後の作業量変化

遅延予測イベント前後における作業量の変化を把握するため，各イベントを基準として，イベント発生日の直前および直後それぞれ 7 日間における 1 日あたり平均作業時間を算出し，前後比較を行った。

表 8 に，遅延予測イベント前後の 7 日平均作業時間の記述統計量を示す。イベント前の平均作業時間は 29.39 分（中央値 8.57 分）であったのに対し，イベント後は 24.36 分（中央値 5.79 分）となり，平均値・中央値ともに低下が確認された。

作業時間の差分（after–before）については，平均値が -6.00 分，中央値が -5.00 分であり，27 件中 73% にあたるイベントにおいて，イベント後の作業量がイベント前を下回っていた。

表 8: 遅延予測イベント前後における 7 日平均作業時間の記述統計（$n = 27$）

	平均	標準偏差	最小	中央値	最大
イベント前（before）	29.39	37.74	0.00	8.57	142.86
イベント後（after）	24.36	35.74	0.00	5.79	125.71
差分（after–before）	-6.00	24.15	-60.00	-5.00	65.71

図 11 に，遅延予測イベント前後の平均作業時間分布を示す。多くのイベントにおいて，イベント後の作業時間がイベント前よりも小さい値を示しており，作業量が減少する傾向が視覚的にも確認できる。

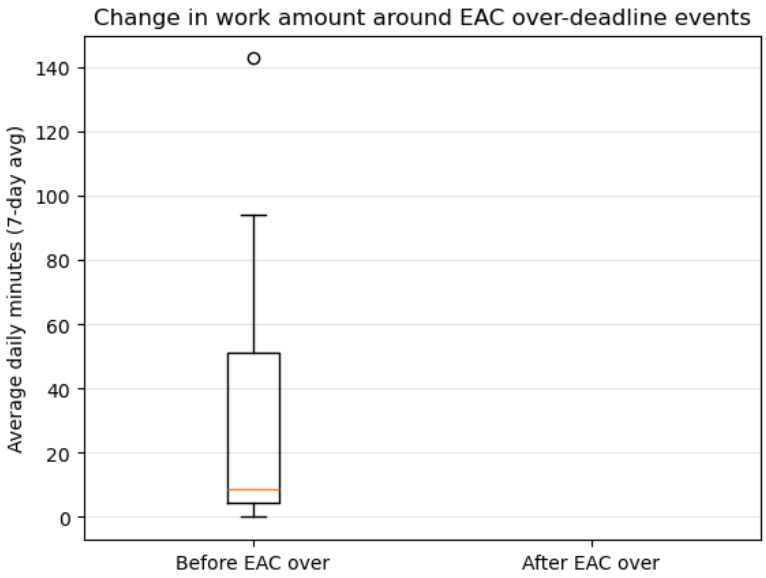

図 11: 遅延予測イベント前後の平均作業時間の分布

6.2.4 ログイン条件別の挙動（補足分析）

本節では，遅延予測イベントが発生した日における利用条件の違いが作業行動に与える影響を補足的に整理する。具体的には，遅延予測イベント当日に利用者のログインが確認された場合と，確認されなかった場合とで，イベント前後の作業量変化を比較する。

表 9 に，遅延予測イベント全体およびログインが確認されたイベントに限定した場合の 7 日平均作業時間の記述統計量を示す。

全 27 件の遅延予測イベントでは，イベント前の平均作業時間が 29.39 分であったのに対し，イベント後は 24.36 分となり，平均差分は -6.11 分であった。中央値においても，イベント前後で 8.57 分から 5.79 分へと低下していた。

一方，ログインが確認された 21 件のイベントに限定すると，イベント前の平均作業時間は 31.93 分，イベント後は 23.08 分となり，平均差分は -8.84 分であった。中央値についても，イベント前後で 9.29 分から 8.57 分へと低下しており，ログイン有りのイベントでは，作業量の減少幅が全体よりも大きい傾向が確認された。

表 9: 遅延予測イベントのログイン有無別記述統計

ラベル	n	平均 before	中央値 before	平均 after	中央値 after	平均差分
全イベント	27	29.39	8.57	24.36	5.79	-6.11
ログイン有り	21	31.93	9.29	23.08	8.57	-8.84

以上より，遅延予測イベント後の作業量は，全体として減少する傾向を示しており，特にログインが確認されたイベントでは，平均的な作業量低下がより大きい値として観測された。

6.2.5 考察

本分析では，将来予測指標である EAC に基づいて締切超過が示唆された遅延予測イベントを基準とし，イベント前後における利用者の作業量変化を観察的に分析した。その結果，多くの遅延予測イベントにおいて，イベント後の作業量は増加するのではなく，むしろ減少する傾向が確認された。

この結果は，EAC による遅延予測の提示が，利用者に対して即時的な作業量増加を促す警告として機能したとは言い難いことを示している。一方で，作業量が一貫して低下している点は，EAC が行動に影響を与えていないことを意味するものではなく，利用者が将来的な完了見通しを再評価し，作業方針や計画そのものを見直した結果として行動が変化した可能性を示唆している。

特に，EAC は現在までの実績ペースに基づいて将来の完了見通しを提示する指標であるため，「このまま作業を続けても締切に間に合わない」という認知を利用者に与えた場合，作業量を一時的に増やすのではなく，タスクの優先順位変更，締切や計画の再設定，あるいは一時的な作業中断といった計画レベルでの行動修正が選択されることも十分に考えられる。本研究のログ設計では，こうした計画再構成行動は作業時間の減少として観測されるため，本分析結果は，行動放棄ではなく合理的な自己調整行動を反映している可能性がある。

また，ログインが確認された遅延予測イベントでは，全体と比較して作業量の低下幅が大きい傾向が観測された。これは，進捗指標や行動支援情報を閲覧可能な状態にあった利用者ほど，EAC に基づく将来予測を踏まえた計画の見直しや判断を行っていた可能性を示唆する結果である。ただし，本研究では利用者が実際に EAC の内容を閲覧したかどうかを直接確認できるログは取得しておらず，EAC の提示と行動変化との因果関係を断定することはできない。

以上より，本分析の結果は，EAC が利用者の作業量を即時的に増加させる指標ではなく，将来の進捗状態を再評価させることで，行動方針や計画の見直しを伴う行動変化と関連している可能性を示している。この点において，短期的な進捗評価に基づき作業行動の調整を促す SPI とは異なり，EAC は中長期的な見通しの再構築を支援する指標として，自己調整学習における役割が異なることが示唆される。

6.3 分析 3：今日のプラン提示と実行行動の整合性

6.3.1 分析の目的

本分析は，第 5 章で設定した研究課題 RQ3「SPI および EAC に基づいて生成される『今日のプラン』の提示は，利用者の当日のタスク選択および作業時間配分にどのような影響を与えるか」を検証することを目的とする。

本研究で実装した「今日のプラン」は，進捗評価指標（SPI）および将来予測指標（EAC）に基づき，当日に取り組むことが推奨されるタスクとその目安時間を提示する行動支援機構である。ただし，本システムは利用者の行動を強制するものではなく，意思決定を支援するための参考情報として提示される設計となっている。

そのため，本分析では，当日プランに含まれるタスクと実際に実行されたタスクとの対応関係，および計画作業時間と実績作業時間との関係に着目し，プラン提示が利用者の行動選択および時間配分にどの程度反映されているかを，実利用ログに基づいて観察的に検証する。

6.3.2 結果

本分析では,「今日のプラン」に含まれるタスクが，利用者の実際の行動選択および作業時間配分にどのように反映されているかを整理した。

まず，当日プランに含まれるタスク（planned task）について，当日に実際に作業が行われた割合（実行率）を算出した。その結果，planned task の実行率は 0.34 であり，当日プランに含まれるタスクのうち，約 34% が当日に実行されていた。

一方，当日プランに含まれないタスク（unplanned task）については，実績作業時間が記録された事例は確認されなかった。すなわち，本分析対象期間においては，プラン外のタスクが自発的に実行されるケースは観測されなかった。

次に，当日に少なくとも 1 件以上の作業実績が記録された日を対象として，実行されたタスクの内訳を分析した。実行タスクが確認された日は 13 日であり，これらの日に実行されたタスクは，すべて当日プランに含まれるタスクであった。したがって，実行されたタスクに占める planned task の割合は 1.00，unplanned task の割合は 0.00 であった。

6.3.3 考察

以上の結果より，SPI および EAC に基づいて生成される「今日のプラン」は，利用者の行動選択に一定程度反映されていることが確認された。特に，実行されたタスクがすべて planned task であった点は，当日プランが「何を行うか」という行動内容の選択に対して，強い影響を与えていることを示唆している。

一方で，planned task の実行率は約 34% にとどまっており，当日プランに含まれているからといって，必ずしも実行されるわけではなかった。この結果は,「今日のプラン」が行動を拘束するものではなく，利用者が当日の状況や判断に応じて，実行可否を選択していることを示している。

また，時間遵守率に大きなばらつきが見られたことから，利用者は当日プランを参照しつつも，作業時間については柔軟に調整していると考えられる。すなわち,「今日のプラン」は作業時間を厳密に規定するものではなく，行動の目安として機能していると解釈できる。

これらの結果から，本システムにおける「今日のプラン」は，行動内容（何を行うか）の選択を支援する一方で，行動量（どれだけ行うか）は利用者の自己調整に委ねる設計として機能していることが示唆される。これは，自己調整学習における主体的な意思決定を阻害せず，判断支援として行動を導くという，本研究の設計思想と整合的な結果である。

6.4 総合考察

本研究の目的は，個人の長期タスクに対して「遅延の早期検知」「行動修正」「進捗理解の促進」を実現する進捗管理モデルを構築することであった。本研究は，進捗指標の精度そのものを評価することを主目的とするのではなく，日次実績ログに基づいて算出・更新される進捗指標や行動提示が，利用者の遅延認知や行動選択にどのように関与し得るかに着目している。以下では，第 5 章で設定した研究課題 RQ1〜RQ3 に対応する分析 1〜3 の結果を統合し，自己調整プロセスの観点から総合的に考察する。

分析 1 では，日次実績入力を契機として更新される進捗評価指標 SPI に着目し，入力前後における ΔSPI の挙動を分析した。その結果，特に尺度の一貫性が保たれる weekly→weekly 条件下において，ΔSPI は正の方向に有意に偏っており，遅延状態からの回復を含む進捗改善が一貫して

観測された。この結果は，遅延閾値の交差といった特定条件に限定されず，日次実績入力と進捗指標更新という行為そのものが，利用者に進捗状態の再認知を促し得ることを示唆している。すなわち，本システムは自己モニタリング（実績入力）を起点として，進捗効率の変化を継続的に可視化することで，日常的な進捗認知を支援する基盤として機能していると考えられる。

分析 2 では，将来予測指標 EAC に基づいて締切超過が示唆された遅延予測イベントを対象とし，イベント前後における作業量の変化を分析した。その結果，多くのイベントにおいて，イベント後の作業量は増加するのではなく，むしろ減少する傾向が観測された。この結果は，EAC が即時的な作業量増加を促す警告として機能したとは言い難い一方で，将来的な完了見通しに関する再評価を利用者に促した可能性を示唆している。特に，EAC は現在までの実績ペースを前提とした予測指標であるため,「現状のままでは達成が困難である」という認知が生じた場合，作業量を一時的に増加させるよりも，タスクの優先順位変更，締切や計画の再設定，あるいは一時的な作業中断といった計画レベルでの調整行動が選択されることも合理的である。本研究のログ設計では，こうした計画再構成行動は作業時間の低下として観測されるため，本分析結果は，行動放棄ではなく，中長期的視点に基づく自己調整行動を反映している可能性がある。一方で，作業量の変化のみから行動の質を区別することは困難であり，この点は本研究の分析上の制約として位置づけられる。

分析 3 では，SPI および EAC に基づいて生成される「今日のプラン」が，利用者の行動選択および時間配分にどの程度反映されているかを検証した。その結果，当日プランに含まれるタスクは実行されやすく，実行が確認された日では，実行タスクがすべて planned task に集中していた。このことは,「今日のプラン」が作業時間を厳密に拘束するものではない一方で，当日に何に着手するかという行動選択の段階において，強い指針として機能していることを示唆している。一方，時間遵守率にはばらつきが見られ，利用者は当日プランを参照しつつも，当日の状況や判断に応じて作業時間を柔軟に調整していた。これは，本システムが行動を強制するのではなく，利用者の裁量を前提とした行動支援として設計されている点と整合的である。

以上の結果を総合すると，本研究で提案したモデルは，日次実績ログに基づく進捗指標の継続的再計算を通じて，遅延状態の認知（SPI），将来リスクの評価（EAC），および行動選択への接続（今日のプラン）を段階的に結び付ける**動的フィードバック枠組み**を構成しているといえる。この枠組みでは，自己モニタリング・自己評価・自己制御という自己調整プロセスが，日次単位で循環的に実行される構造としてシステム上に具現化されている。

7　まとめ

本研究では，プロジェクトマネジメント分野で用いられてきた Earned Value Management（EVM）理論を基盤として，日次実績ログに基づく個人タスク進捗管理モデルの提案および実装を行った。従来の ToDo 管理やリマインド型支援が，締切直前の気づきや主観的な進捗感に依存しがちであったのに対し，本研究は，日次実績の蓄積と指標更新を通じて，進捗状態を継続的に可視化し，利用者の自己調整を支援する枠組みの構築を目的とした。

本研究の特徴は，EVM 理論を単に個人タスクへ適用するのではなく，**日次実績ログという粒度に再構成し，自己調整プロセスの一部として再解釈した点**にある。企業プロジェクトを前提とする従来の EVM は，固定的な計画と管理者による評価を想定しているが，個人タスクでは，計画の揺らぎや主観的判断が避けられない。本研究では，こうした特性を前提とした上で，SPI を「進捗効率の再認知」，EAC を「将来リスクの再評価」,「今日のプラン」を「行動選択の支援」と位置づけ，それぞれを日次で循環させるモデルとして実装した。

実利用ログを用いた分析の結果，SPI は遅延状態や進捗変化の認知を日常的に促す指標として機能し，実績入力を通じた進捗状態の更新が，継続的な自己モニタリングを支えていることが確認された。一方で，EAC に基づく将来予測は，即時的な作業量増加を直接促すものではなく，計画の見直しや方針転換といった中長期的な自己評価を喚起する指標として機能することが示された。さらに，SPI および EAC に基づいて生成される「今日のプラン」は，行動を強制するものではないが，提示されたタスクが当日の実行対象として選択される傾向が確認されており，利用者が何に着手するかという行動選択の基準として機能していることが示された。

これらの結果を総合すると，本研究で提案・実装した進捗管理モデルは，日次実績入力を起点として，進捗評価，行動提示，実行が循環的に更新される**動的フィードバック枠組み**を構成しているといえる。本枠組みでは，自己モニタリング（実績の記録），自己評価（SPI・EAC による進捗把握），自己制御（行動選択・計画調整）が，日次単位で連続的に結び付けられている。図 12 は，本研究で提案したモデルを，自己調整プロセスの観点から整理した概念図である。

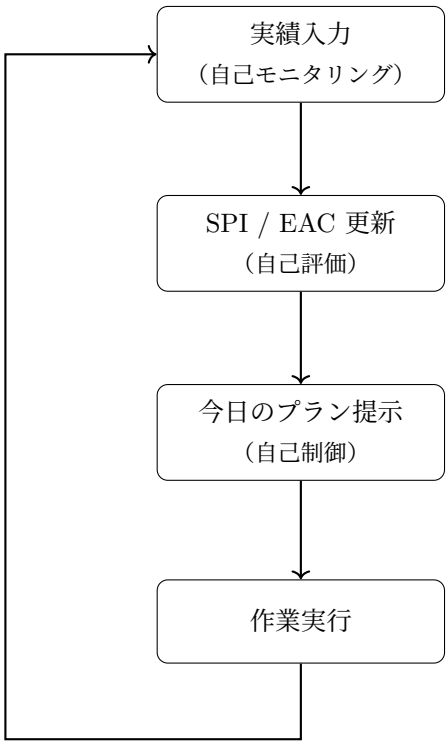

図 12: 日次実績ログに基づく自己調整プロセスを実装した動的フィードバック枠組み

本研究の結果は，EVM 理論を個人タスク管理へ応用する際には，指標の精度や達成率のみを追求するのではなく，**進捗を「どのように認知し，どのように判断し，どのように行動に結び付けるか」**という自己調整過程全体を支援する視点が重要であることを示している。特に，日次実績ログに基づく継続的な指標更新は，締切直前に遅延へ気づく従来の管理手法とは異なり，進捗認知と将来評価を早期に生起させる点で，個人タスクに適した支援構造を形成していると考えられる。

一方で，本研究の分析には方法論的な制約が存在する。本研究では，SPI や EAC に基づく進捗リスクの提示や「今日のプラン」による行動支援と，その後の作業行動との関係を実利用ログに基づいて分析したが，利用者がこれらの警告や提示画面を実際に「閲覧・認知」したかどうかをログ上で厳密に区別することはできていない。そのため，本研究で観測された行動変化は，進捗情報の

提示そのものによる影響を直接的に測定した結果ではなく，提示と認知が生じた可能性を含むものとして解釈する必要がある。今後は，警告表示やプラン画面の閲覧ログを取得し，進捗認知の有無を考慮した分析を行うことで，指標提示と行動変化の関係をより厳密に検証することが求められる。

また，EAC による将来リスクの提示が，必ずしも作業量の増加に直結しない点は，本モデルにおける重要な検討課題である。将来予測の提示が，合理的な計画再構成を促す場合と，過度な行動抑制や意欲低下につながる場合をどのように区別し，適切な行動支援へ接続するかは，今後の研究課題である。加えて，タスクの再設定や優先順位変更といった計画レベルの行動を定量的に捉える指標設計についても，さらなる検討の余地がある。

以上より，本研究は，EVM 理論を日次実績ログという粒度に再構成し，個人タスクにおける自己調整プロセスを動的に支援する進捗管理モデルを提案・実装した。さらに，実利用ログに基づく分析を通じて，本モデルが進捗認知および行動選択を支援する動的フィードバック枠組みとして機能し得ることを検証した。本成果は，個人の長期タスク管理において，自己調整学習を支援する新たな進捗管理手法の基盤となり得ると考えられる。

参考文献

[1] Ziółkowska, A., & Połoński, M. (2016). Application of the EVM method and its extensions in the implementation of construction objects. Engineering Structures and Technologies, 7(4), 189–196. (EVM が最終的な期間とコストの予測を可能にする有効なツールであることを示した文献)

[2] 野上 俊一, 生田 淳一, 丸野 俊一,“テスト勉強の学習計画と実際の学習活動とのズレに対する認識”, 日本教育工学会論文誌, 28(suppl), pp. 173–176, 2005.

[3] Project Management Institute, “A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Seventh Edition”, Project Management Institute, 発行年:2021.

[4] Zimmerman, B. J. (2002). Becoming a self-regulated learner: An overview. Theory into practice, 41(2), 64–70. (自己調整学習（Self-regulation）の循環モデルの基本的な枠組みを裏付ける文献)

[5] DiBenedetto, M. K., & Zimmerman, B. J. (2010). Differences in self-regulatory processes among students studying science: A microanalytic investigation. The International Journal of Educational and Psychological Assessment, 5(1), 1–17. (SRL の自己評価とモニタリングが学業成績に相関することを実証的に示した文献)

[6] Morita, D., & Suwa, H. (2012). An Activity Duration Estimation Method Aiming at Enhancing Schedule Stability in Project Management. Transactions of the Japanese Society for Mechanical Engineers, 25(6), 145–151. (作業見積もりにおける不確実性とスケジュール安定性の重要性を示した文献)

[7] NASA, “Earned Value Management (EVM) Implementation Handbook”, NASA, 発行年:2019.

[8] U.S. Department of Defense, “Earned Value Management System (EVMS) Guidelines”, DoD, 発行年:2018.

A　進捗指標計算アルゴリズムの詳細

本付録では，本文 4.4.4 節で定義した進捗指標（SPI，EAC，pace7d，requiredPace 等）について，updateStats.js においてどのような手順および条件分岐で計算・更新しているかを示す。

各指標の数式定義および例外条件については，本文 4.4.4 節に集約されているため，本付録では数式の再掲は行わず，実装上の処理フローに焦点を当てて説明する。

A.1　アルゴリズムの位置づけ

updateStats.js は，日次実績ログの追加・編集を契機として実行され，対象タスクに関する進捗指標を再計算する中核的な処理である。

本アルゴリズムは，単一の計画値を前提とする静的な進捗管理ではなく，実績入力のたびに全指標を更新する動的 EVM の実装として設計されている。

A.2　再計算トリガ

進捗指標の再計算は，以下のいずれかのイベントを契機として実行される。

- 日次実績ログが新たに追加された場合
- 既存の日次実績ログが編集された場合

これにより，過去ログの修正や入力遅延が生じた場合でも，常に最新の実績に基づいた進捗評価が行われる。

A.3　処理全体の流れ

updateStats.js における進捗指標計算は，以下の順序で実行される。

1. 対象タスクに紐づく全日次実績ログの取得
2. 累積実績および残作業量の更新
3. 締切までの残日数の算出
4. 直近作業ペース（pace7d）の算出
5. 必要ペースおよび調整後必要ペースの更新
6. 進捗効率指数（SPI）の算出
7. 条件を満たす場合に完了予測日（EAC）を更新

各段階の計算結果は，後続の指標算出に用いられる。

A.4　直近作業ペース算出時の補正処理

pace7d の算出においては，利用初期における評価の不安定性を考慮し，実績が十分に蓄積されていない場合の補正処理を行う。

具体的には，直近 7 日間に実績が存在する日数が一定未満の場合には，固定日数による平均化を行わず，実績が存在する日数を分母として平均ペースを算出する。

これにより，初期段階における過小評価を防ぎつつ，実績の蓄積に伴って安定した進捗評価へと移行する。

A.5　進捗指標の更新条件

以下の場合には，一部またはすべての進捗指標の更新を行わない。

- 直近作業ペースが算出できない場合
- 計画開始日が未到達である場合

これらの条件は，進捗評価が意味を持たない状態での誤った指標提示を避けるために設けられている。

A.6　完了予測日（EAC）の扱い

完了予測日（EAC）は，算出条件を満たす場合にのみ更新される。

算出条件の詳細は本文 4.4.4 を参照し，本付録では更新順序と条件分岐のみを扱う。

A.7　付録としての役割

以上のように，本付録では進捗指標の計算処理について，実装上の手順および条件分岐の流れを整理した。

数式定義および指標の理論的意味については，本文 4.4.4 節に集約されており，本付録はそれらを補足する実装仕様の理解を目的とした資料として位置づけられる。

B　当日の作業計画におけるタスク選定アルゴリズム

本付録では，第 4 章で概要を述べた「今日のプラン」機能において用いられる，当日の作業計画（最大 3 件）を生成するアルゴリズムの実装仕様を示す。本アルゴリズムは，進捗指標および締切情報を統合的に参照し，候補タスクの優先度評価と日次キャパシティに基づく作業分数の配分を行う。

本アルゴリズムは，フロントエンドおよびバックエンドの双方から利用可能な共通モジュールとして実装されており，入力情報が同一であれば実行環境に依存しない一貫した判定結果が得られる構成としている。

B.1　アルゴリズムの入力と出力

入力は，各タスク i に対して以下の情報である。

- 見積作業時間 E_i［分］
- 累積実績時間 A_i［分］
- 残作業量 $R_i = E_i - A_i$［分］
- 締切日時 $deadline_i$
- 予定開始日時 $plannedStart_i$
- 必要ペース［分/日］（調整後必要ペース `requiredPaceAdj`，または基本必要ペース `requiredPace`）
- 理想進捗率および実進捗率
- ユーザー設定の日次キャパシティ C［分］

出力は，当日に取り組むタスク集合 $\mathcal{P}$ と，各タスクに割り当てられる推奨作業時間（`todayMinutes`）である。キャパシティ制約により割当が 0 分となるタスクは除外されるため，$|\mathcal{P}|$ は最大 3 件であるが，状況によっては 3 件未満となる場合がある。

B.2　日次キャパシティの決定

当日に利用可能な作業時間の上限を日次キャパシティ C［分］とする。C が未設定，0 以下，または不正値である場合には，安全策として既定値 $C = 120$ 分を用いる。

実装上は再利用性の観点から一時的なキャパシティ指定を受け付ける設計としているが，本研究における「今日のプラン」の算出および再計算では，原則としてこの日次キャパシティを上限として処理を行う。

B.3　候補タスク集合の定義

各タスク i について，以下の条件をすべて満たすものを候補タスク集合 $\mathcal{S}$ に含める。

- 予定開始日時 $plannedStart_i$ が現在時刻以前である
- 見積作業時間 $E_i > 0$
- 累積実績時間 $A_i < E_i$（未完了）
- 締切日時 $deadline_i$ が設定され，日付として解釈可能である

このように，遅延状態にあるタスクのみを対象とするのではなく，締切が近づいている進捗良好なタスクも候補に含めることで，締切直前の未着手を防ぐ設計としている。

B.4　遅れ度および残日数の定義

候補タスク $i \in \mathcal{S}$ について，理想進捗率および実進捗率をそれぞれ

$$\text{ideal}_i = \frac{t_i}{T_i}, \qquad \text{actual}_i = \frac{A_i}{E_i}$$

と定義する。ここで，t_i はタスク開始から現在までの経過時間，T_i はタスク全体の計画期間を表す。

遅れ度 lag_i は，

$$\text{lag}_i = \text{ideal}_i - \text{actual}_i$$

と定義する。優先度評価では進捗超過を重視しないため，$\max(0, \text{lag}_i)$ を用いる。

締切までの残日数 D_i は，

$$D_i = \max\left(1, \left\lceil \frac{deadline_i - now}{1\,day} \right\rceil\right)$$

とし，当日締切の場合も $D_i = 1$ とする。

B.5　優先度スコアの算出

候補タスク i の優先度は，遅れ度，締切リスク，および残作業負荷を統合した優先度スコア S_i により評価する。

$$S_i = 3 \cdot \max(0, \text{lag}_i) + 2 \cdot \frac{1}{D_i + 1} + \frac{R_i}{D_i}$$

ここで，本実装では進捗遅れ（lag_i）を最も重視し，次いで締切の近さ，残作業量に基づく負荷を考慮する設計とした。係数 3 および 2 は，進捗遅れを優先的に回復させるという設計思想にもとづき，経験的に設定した重みである。

候補集合 $\mathcal{S}$ は S_i の降順に整列される。同点の場合は，締切が近い順，次いで必要ペースが大きい順に決定する。

B.6　日次キャパシティに基づく割当

優先度順に整列された候補タスクに対し，日次キャパシティ C の範囲内で当日の推奨作業時間を割り当てる。割当は，以下の三段階で実行される。

1. **最低ライン（必須割当）**：締切までに間に合わせるために最低限必要な作業量を割り当てる。
2. **回復ライン（進捗改善）**：進捗遅れの改善を目的とし，進捗効率指数（SPI）を計画通りの状態（SPI = 1.0）へ回復させることを目標とした追加割当を行う。
3. **余剰キャパシティの活用**：上記を満たした後にキャパシティが残る場合，締切が近い順に前倒し的な割当を行う。この余剰配分は，1 タスクあたり最大 30 分を上限とする。

割当結果が 0 分となるタスクは，当日の作業計画から除外される。

B.7　フォールバック処理

上記の割当処理を経ても当日の作業計画 $\mathcal{P}$ が空となる場合に限り，フォールバック処理を行う。

この場合，候補集合 $\mathcal{S}$ の整列結果を用いて最大 3 件のタスクを対象とし，残作業量を締切までの残日数で日割り計算した作業量を割り当てる。日割り計算は 5 分単位で切り上げて行い，キャパシティを超えない範囲で割り当てる。

B.8　疑似コード

```
function generateDailyPlan(tasks, userSettings, now):

  C = resolveDailyCap(userSettings)

  S = extractCandidateTasks(
        tasks,
        plannedStart <= now,
        E > 0,
        A < E,
        valid deadline
      )

  for task in S:
    compute lag, D, R
    compute score S_i

  sort S by (S_i desc, deadline asc, requiredPace desc)

  P = allocateTodayMinutes(S, C)
    - pass1: allocate minimum required work
    - pass2: grow toward recovery target (SPI = 1.0)
    - pass3: distribute surplus by nearest deadlines
             (max +30 min per task)

  remove tasks with todayMinutes == 0

  if P is empty and S not empty:
    P = fallbackAllocateTop3(S)  // per-day split, ceil to 5 min

  return P
```