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 2 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，本研究で扱うような進捗効率や将来予測を，日次の実績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。

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扱える点に着目し，EVM の考え方を個人タスク管理へ応用する。

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，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分析を通じて，遅延の早期検知，行動修正，および進捗理解の促進という観点から，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・EAC 超過提示後の作業量変化や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（本研究では SPI < 0.8）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。
- **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 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。

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）

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，$\mathrm{SPI} \geq 1.0$ を計画通り，$0.8 \leq \mathrm{SPI} < 1.0$ を軽度の遅延，$\mathrm{SPI} < 0.8$ 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。

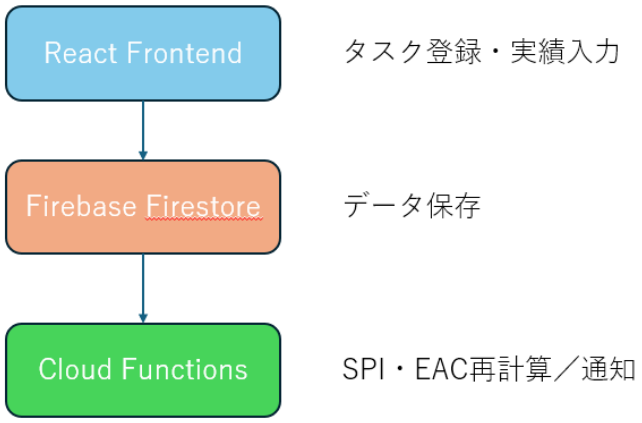

図 1: 本研究システムの全体構成

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。

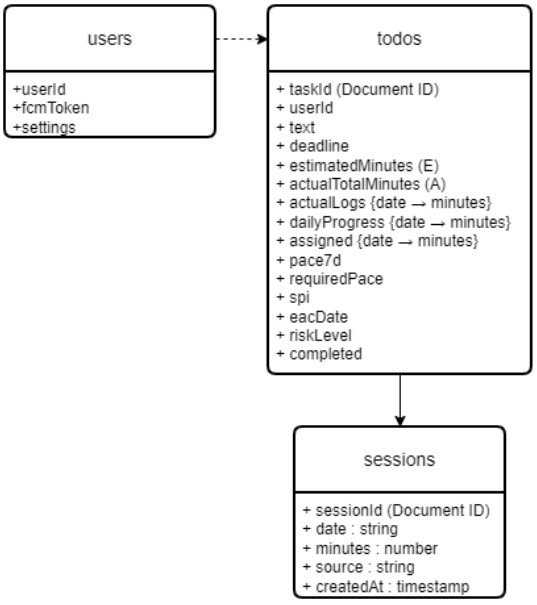

図 2: Firestore のデータモデル（タスク・実績ログ）

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。

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。

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。

4.4.2 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 `actualLogs` が変更されるたびに，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，日次実績ログに基づき，タスクの進捗状態を継続的に更新するため，updateStats.js に指標計算処理を集約した。本機能により，ユーザーの実績入力や過去ログの修正に応じて，Firestore 上の指標値が常に最新状態となるよう維持される。

本研究で updateStats.js が再計算対象とする主な指標を以下に示す。

- 残作業量：$R = E - A$
- 直近平均ペース：pace7d（過去 7 日間の平均作業時間）
- 必要ペース：requiredPaceAdj（締切までに必要な 1 日あたり作業量）
- 進捗効率指数：$\mathrm{SPI} = \frac{\mathrm{pace7d}}{\mathrm{requiredPaceAdj}}$

- 予測完了日（EAC）：現在の pace7d を継続した場合の完了見込み日

計算後，Cloud Functions は各指標を Firestore の該当フィールドに書き戻し，フロントエンドの表示および後続処理で参照可能な状態とする。初期 3 日間は pace7d の分母を「作業があった日数」で割るウォームアップを行い，SPI（warn しきい値 0.8）と EAC による遅延判定を優先する。締切超過または EAC 超過時は SPI に関係なく遅延とし，リスク分類は ok / warn / late の 3 段階で表示する。

EAC 予測日の算出と履歴的可視化　本研究では，EAC を単一の固定値として保持するのではなく，日次の実績ログに基づいて動的に再計算される予測完了日として扱う。updateStats.js は，実績ログ（`actualLogs`）が更新されるたびに，その時点の実績ペースに基づいて 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。

EAC の時系列的な変化については，Firestore に履歴として永続化するのではなく，日次実績ログを基にフロントエンドおよび分析処理において必要に応じて再計算・可視化する方式を採用している。この「再計算可能性を重視した設計」により，過去ログの編集や修正が行われた場合でも，全期間にわたる EAC の整合性を常に保った状態で再評価を行うことが可能となる。

このように，本研究では，EAC の履歴管理を「値の保存」ではなく「日次ログに基づく再計算可能性」によって担保する設計とし，動的な進捗評価を重視した。

4.4.4　リスクレベル判定（SPI 基準）

本システムでは，進捗指標として算出される SPI の値に基づき，タスクの進捗状態を段階的に分類する。SPI の値は Cloud Functions 内の判定ロジックで参照され，以下の 3 段階のリスクレベルとして整理される。

- SPI ≥ 1.0：良好
- $0.8 \leq$ SPI < 1.0：注意
- SPI < 0.8：遅延

また，EAC が締切日時を超過すると判定された場合や，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参照される。

4.4.5　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 A において詳述する。

4.4.6　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。

4.4.7　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。

4.4.8　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。

4.4.9　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。

4.4.10　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。4.4.8 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。

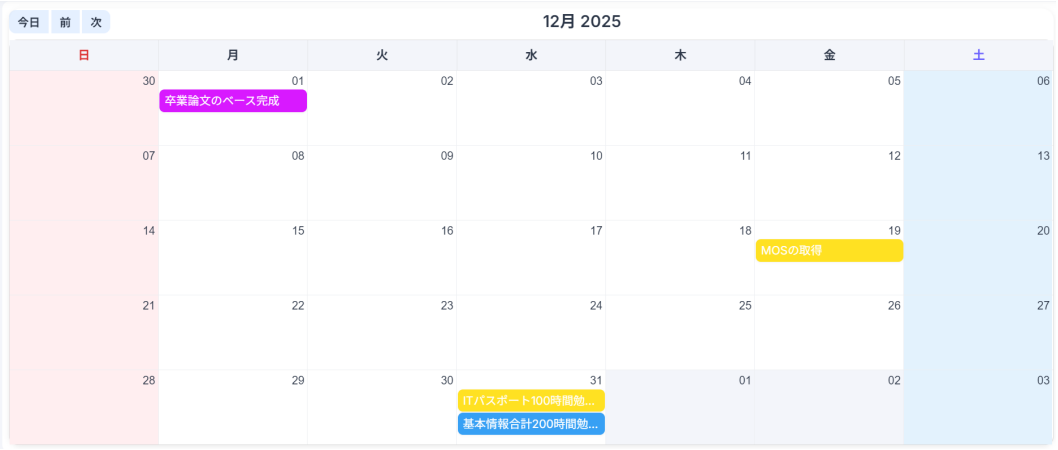

図 3: カレンダー画面（月間のタスク進捗と締切の俯瞰）

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。

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

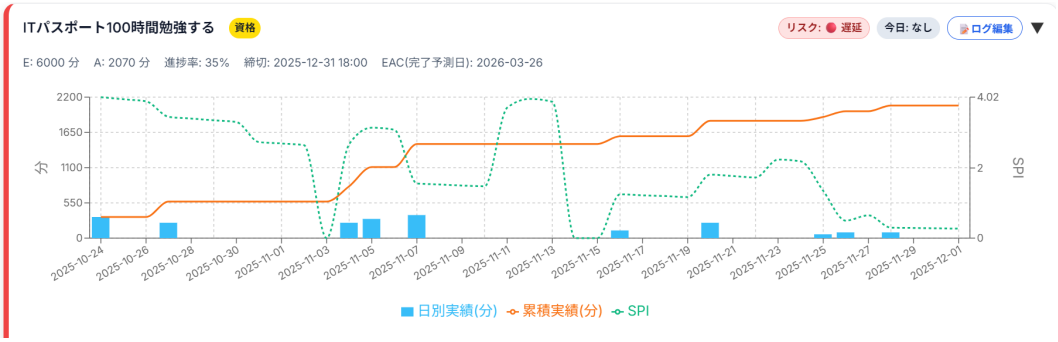

図 5: 分析画面（作業時間推移，直近平均，SPI 推移）

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。

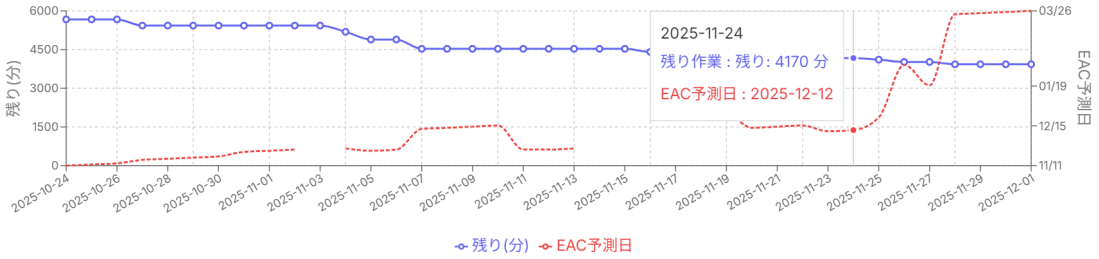

図 6: EAC 予測日の時系列グラフ（残作業量との比較）

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

図 7: ログ編集画面（入力忘れや誤入力の修正に対応）

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。

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

図 8: 通知設定画面

- 残作業量 R
- 必要ペース（requiredPaceAdj）
- 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 4.4.7 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

5.1　実験の目的

本実験の目的は，日次で記録された実利用ログを用いて，進捗評価指標および行動提示が，利用者の行動とどのような関係を持つかを検証することである。

本研究では，利用者が進捗の遅延状態や将来的な締切超過リスクを認知した際に，作業量や行動選択に変化が生じているかに着目し，提案する進捗管理システムの有効性を評価する。

具体的には，以下の三つの観点から分析を行う。

第一に，進捗評価指標に基づいて遅延状態が示された際に，その前後で作業量や作業ペースに変化が生じているかを検証する。

第二に，将来的な締切超過が予測された際に，その前後で利用者の作業行動に変化が生じているかを検証する。

第三に，当日提示された行動計画と，実際に選択・実行された行動との対応関係を分析し，進捗評価に基づく行動提示が，行動選択および時間配分とどのように関係しているかを検証する。

5.2　データセットと観測期間

本研究では，開発したアプリの実利用ログを用いた観察研究として分析を行う。解析に用いる主なデータは以下のとおりである（いずれも匿名化済みユーザ ID を含む）。

- 日次実績ログ（`daily_logs.csv`）：229 レコード，7 ユーザ，期間 2025/04/08–2025/12/26。
- タスク実績ログ（`todos_logs_export_20251215_054532.csv`）：92 レコード，6 ユーザ，期間 2025/10/15–2025/12/15。
- 今日のプラン提示ログ（`dailyPlans_export_20251209_044520.csv`）：9 レコード，1 ユーザ，期間 2025/12/09 のみ。

日次実績ログが全分析の基盤となるが,「今日のプラン」に関するデータは 1 ユーザ・9 件と小規模であるため，RQ3 の解釈には慎重を期す。データ取得時点での欠損や異常値（負の時間，締切の欠落）は除外し，時刻は日単位に正規化して集計する。また，分析 1 では「締切日以降の投入」と「締切日と完了日が同日の投入」（締切ゆえの着手で SPI に基づく行動と解釈しにくい）を除外し，分析対象ログは 229 件→ 186 件となる。日次実績ログが全分析の基盤となるが,「今日のプラン」に関するデータは 1 ユーザ・9 件と小規模であるため，RQ3 の解釈には慎重を期す。データ取得時点での欠損や異常値（負の時間，締切の欠落）は除外し，時刻は日単位に正規化して集計する。また，分析 1 では「締切日以降の投入」と「締切日と完了日が同日の投入」（締切ゆえの着手で SPI に基づく行動と解釈しにくい）を除外し，分析対象ログは 229 件→ 186 件となる。ログイン状態（`loginFlag`）は通知閲覧の近似として併記し，後述の分析 1 でログイン有無別の比較も行う。

5.3　評価指標と分析手順

各研究課題に対応する主要指標を以下に定める。

- RQ1（遅延認知と行動変化）：日次の実績入力により $\Delta\text{SPI} \neq 0$ となった入力イベントを起点とし，入力前後の SPI と pace7d の変化量（例：ΔSPI，Δpace7d）を比較することで，進捗状態の更新が行動に与える影響を評価する。
- RQ2（将来リスク認知と行動変化）：EAC が締切を超過した日を起点に，翌 7 日間の「EAC 超過日数」と「調整後必要ペース（requiredPaceAdj）」の変化量を評価する。
- RQ3（行動提示と実行）：当日提示された todayMinutes と，当日の実績投入時間との差分 $|A_{\text{day}} - \text{todayMinutes}|$ を算出し，その差分と翌日の SPI 変化量との相関を分析する（データは 1 ユーザ・9 件と極小であり，解釈は探索的に留める）。

集計はユーザ単位で行い，複数イベントが同日に重なる場合は最新の指標を採用する。閾値交差イベントが複数回発生した場合は，7 日以上離れたイベントのみを独立とみなす。

5.4 ベースラインおよび比較条件

提案手法（EVM 指標提示＋行動提示）との比較基準として，以下の条件を設定する。

- **非介入区間（プレ警告期）**：同一ユーザの警告発生前 7 日間をベースラインとし，警告後と比較する。
- **EAC 非提示区間**: EAC が締切内に収まっている期間を対照とし，EAC 超過提示後と比較する。
- **行動提示なし（プラン未生成日）**：今日のプランが生成されなかった日を対照とし，生成日との乖離を比較する（ただしサンプル数が極小であることを明記）。

5.5 統計的検定と効果量

サンプル数が大きくないため，正規性を仮定しないノンパラメトリック検定を基本とする。

- 事前・事後のペア差分に対して Wilcoxon の符号付順位検定を適用し，有意水準は $p < 0.05$ とする。
- 効果量は Cliff の δ または相関係数 r を併記し，95% 信頼区間はブートストラップ（5,000 サンプル）で算出する。
- サンプル数が極小（RQ3）である場合は，推定値と信頼区間のみを報告し，統計的有意性の解釈を避ける。

平日/休日，期末前後といった交絡要因については層別集計を行い，効果の頑健性を確認する。期間が短いユーザは記述統計のみに留め，統計検定の対象から除外する。

以上の分析を通じて，進捗評価および行動提示が，利用者の行動とどのように関連しているかを，定量的な実利用ログに基づいて明らかにすることを目的とする。

本章のまとめと次章への接続: データ抽出条件，指標計算，検定方針，ベースラインの定義を明確化し，再現性と解釈範囲の前提を示した。次章では，これらの前提に基づく結果と統計的検証を報告する。

5.6　評価対象および実験条件

本研究で用いたログデータは，5.2 節で示した 7〜6 ユーザ分の実利用ログである（ユーザ ID は匿名化済み）。統制的なタスク割当は行わず，日常利用の中で得られた行動ログを観察する自然実験に近い設計とした。

分析方針は以下のとおりである。

- ユーザ内比較を優先：警告前後や提示前後の変化を，同一ユーザ内でペア比較する。
- ケーススタディ的解釈：サンプルが小規模な指標（特に RQ3）は効果量と信頼区間の報告に留め，過度な一般化を避ける。
- 観察ログの品質管理：負の時間値や締切欠落を除外し，日付単位に正規化したうえで集計する。

利用者は学習・資格試験対策など中長期タスクを自律的に登録し，見積時間と日次実績を入力した。システム側から作業量を強制することはなく，進捗指標と行動提示は閲覧自由とした。タスクごとの属性として設定された見積時間と締切日は，遅延判定や必要ペース算出に利用する。

5.7　分析指標および評価方法

本節では，収集した実利用ログを用いて，提案システムが利用者の進捗認知および行動とどのように関係しているかを検証するための，分析指標および評価方法を定義する。前節で示した 3 つの観点を，以下では分析 1〜3 として整理する。

本研究では，進捗評価指標として算出される SPI および EAC，ならびにそれらに基づいて提示される「今日のプラン」を分析対象とし，日次で記録された実績ログを用いて，指標変化が観測されたタイミングを基準に前後の行動指標を比較する。

5.7.1　分析 1：SPI に基づく遅延認知と行動変化

分析 1 では，進捗評価指標である SPI の**日次更新そのもの**に着目し，実績入力に伴う進捗状態の再認知が，その後の行動変化とどのように関連するかを検証する。

本分析では，日次の実績入力により SPI が再計算され，$\Delta\mathrm{SPI} \neq 0$ となったすべての入力イベントを**進捗状態の更新に伴う認知イベント**として扱う。ここでの認知には，$\mathrm{SPI} \leq 1$ という遅延状態の把握に加え，ΔSPI による改善・悪化方向の変化そのものを含める。

各イベントについて，実績入力前後の SPI を対応付け，その差分を算出することで，指標変化と行動の関係を評価する。

5.7.2　分析 2：EAC に基づく将来予測と行動変化

分析 2 では，将来予測指標である EAC（Estimate at Completion）に着目し，EAC によってタスクの締切超過が予測された際に，利用者の行動に変化が生じているかを検証する。

本研究では，日次で再計算される EAC に基づき，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**として定義する。この遅延予測イベントは，将来的な進捗リスクが顕在化したタイミングとして位置づけられる。

各遅延予測イベントについて，イベント発生日の直前および直後の一定期間における利用者の行動を比較する。具体的には，遅延予測イベントの前後それぞれについて，1 日あたりの平均作業時間および作業頻度を算出し，将来予測の提示前後での行動指標の変化を評価する。

比較は，同一イベント内における前後対応を持つデータとして扱い，前後差に基づく分析を行う。また，前後差の分布特性に応じて，対応のある統計的検定を用いることで，観測された行動変化が偶然によるものか否かを検証する。

本分析により，EAC による将来予測の提示が，自己調整学習における Self-evaluation（将来状態の評価）を通じて，利用者の行動とどのように関連しているかを明らかにする。

5.7.3　分析 3：今日のプラン遵守度の分析

分析 3 では，SPI や EAC に基づいて生成される「今日のプラン」が，利用者の実際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する。

当日提示されたタスクと実行されたタスクの対応関係，ならびに計画作業時間と実績作業時間の関係を用いて，行動提示と実行行動との整合性を評価する。

5.8　評価モードと表記の定義

本研究では，実績ログの蓄積期間に応じて二つの評価モードを用いる。

short 評価モードは，実績ログが 3 日以内の場合に用いられ，7 日移動平均ペース（pace7d）が安定して算出できない状況を想定する。このモードでは，直近数日間の実績ペースを用いて SPI を算出し，着手直後や短期的な行動変化を暫定的に評価する。

weekly 評価モードは，実績ログが 3 日を超えて蓄積された後に用いられ，直近 7 日間の平均作業ペース（pace7d）に基づいて SPI を算出する。週間単位での安定した進捗認知や行動変化を捉える目的で，主にこの weekly 評価モードを用いる。

また,「A → B」という表記は，ある評価時点における評価モード A の進捗状態から，次の評価タイミングにおいて評価モード B によって再評価された状態への遷移を表す。たとえば「weekly → weekly」は，ある週における週間評価結果と，次回の週間評価時点における結果との変化を示すものであり,「short → weekly」は，短期的な進捗変化が観測された後に，次の週間評価においてどのような状態に遷移したかを示している。

6　結果・考察

本章では，前章で定義した評価モードと表記に基づき，分析結果を報告し考察する。

6.1　分析 1：週間評価フェーズにおける SPI 改善効果の検証

本節では，日次の実績入力によって SPI が更新され $\Delta\text{SPI} \neq 0$ となったすべての入力イベントを対象とし，入力前後の進捗状態を比較する。遅延状態（$\text{SPI} \leq 1$）の認知に加えて，更新により変化した方向（改善・悪化）を示す ΔSPI も利用者が受け取る進捗認知の一部として解釈する。

表 4: 日次作業ログにおける記述統計量

	minutes	estimateMinutes
count	186.000	186.000
mean	139.946	6676.129
std	126.907	6625.369
min	0.000	20.000
25%	30.000	247.500
50%	120.000	4000.000
75%	180.000	15000.000
max	1000.000	15000.000

記号の定義と ΔSPI の扱い　本分析で扱う ΔSPI は，実績入力直前の値 $\mathrm{SPI}_{\mathrm{before}}$ と，入力後に当日分を含めて再計算した値 $\mathrm{SPI}_{\mathrm{after}}$ の差分 $\Delta\mathrm{SPI} = \mathrm{SPI}_{\mathrm{after}} - \mathrm{SPI}_{\mathrm{before}}$ と定義する。SPI は required（当日朝時点の必要ペース）に対する 7 日間平均ペース（pace7d）の比であり，weekly モードでは required が入力前後で一定のまま，pace7d に当日実績が加算される。このため weekly→weekly のイベントでは，実績を投入すると SPI が非減少になりやすく，ΔSPI は概ね非負となる（微小な負値は丸め誤差の影響と考えられる）。一方で，締切日以降や締切日と完了日が同日の投入は，締切ゆえの着手と解釈でき，SPI に基づく行動変化を評価するには適さないため，分析 1 の抽出段階で除外した。短期モードとの切替（short→weekly など）では分母となる期間が変わるため，負の値も生じ得る。図 10 のように正方向の値が多い理由は，以上の算出手順とフィルタ条件（$\Delta\mathrm{SPI} \neq 0$）によるものであり，負の値を意図的に除外したわけではない。

6.1.1　データ概要

本分析では，提案システムの実利用により記録された日次作業ログを用いた。実験期間中に収集された日次ログは 229 件であり，対象タスク数は 60 件であった。

6.1.2　日次作業量および推定作業量の分布

表 4 に，分析対象（締切超過・締切日完了の除外後）における実績作業時間（minutes）および推定作業時間（estimateMinutes）の記述統計量を示す。実績作業時間の中央値は 120 分（第 1 四分位 30 分，第 3 四分位 180 分）であり，最大値は 1000 分であった。推定作業時間の中央値は 4000 分（第 1 四分位 247.5 分，第 3 四分位 15000 分）であった。以上より，本研究で扱うログには，日次作業量のばらつきとタスク規模の多様性が含まれていることが確認された。

6.1.3　入力イベントおよび評価モード遷移の内訳

実績入力により SPI が変化した日を入力イベントとして抽出したところ，日次ログ 186 件のうち 178 件が入力イベントとして抽出された。さらに，評価モードの遷移に基づき入力イベントを分類した結果を表 5 に示す。尺度の一貫性が保たれる weekly→weekly を主対象とし，short→short および short→weekly は参考として扱った。

表 5: 入力イベントの評価モード遷移ごとの件数

モード遷移（before→after）	件数
weekly→weekly	109
short→short	53
short→weekly	16

表 6: ΔSPI のモード別集計と Wilcoxon 検定結果

モード遷移	件数	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)

6.1.4 モード別の ΔSPI 傾向

モード別に ΔSPI の偏りを集計した結果を表 6 に示す。weekly→weekly（$n = 109$）では ΔSPI の正方向が 0.954，中央値は 0.324（平均 0.730）であった。short→short（$n = 53$）では中央値 1.50（平均 1.89）とばらつきが大きく，short→weekly（$n = 16$）では中央値が -0.981 と負方向に偏っていた。

weekly→weekly における ΔSPI の分布を図 10 に，SPI の入力前後比較を図 11 に示す。

6.1.5 統計検定による ΔSPI の偏りの検証

weekly→weekly（$n = 120$）に対して，ΔSPI が正の方向に偏っているかを検証するため，Wilcoxon の符号付順位検定（片側検定）を行った。表 6 のとおり $p = 3.46 \times 10^{-13}$，効果量 $r = 0.66$ となり，改善方向への偏りが確認された。

6.1.6 短期評価およびモード遷移ケースの特徴

short→short（$n = 81$）では ΔSPI の分散が大きく，極端な値も観測された（図 12）。また short→weekly のような評価モード遷移を伴うケースでは，分母となる期間の切替により ΔSPI が負方向に振れやすい。入力前後差に算出ルールの影響が含まれ得るため，差分解釈には注意が必要である。以上より，本分析では weekly→weekly を主対象として解釈することが妥当である。

6.1.7 $SPI_{\mathbf{after}} \geq 1$ に到達したイベントの特徴と到達率

実績入力後に $SPI_{\mathrm{after}} \geq 1$ を満たしたイベントを表 7 に示す。weekly→weekly では 109 件中 89 件（中央値 $\Delta\mathrm{SPI} = 0.38$），short→short では 53 件中 44 件（中央値 1.71）が到達し，short→weekly は 16 件中 6 件（中央値 -0.80）であった。

次に，入力後に $SPI_{\mathrm{after}} \geq 1$ を満たした割合（到達率）を表 8 に示す。weekly→weekly の到達率は 0.817 と最も高く，尺度の一貫性が保たれる条件下では，入力後に進捗評価が改善方向へ変化するケースが多いことが示された。

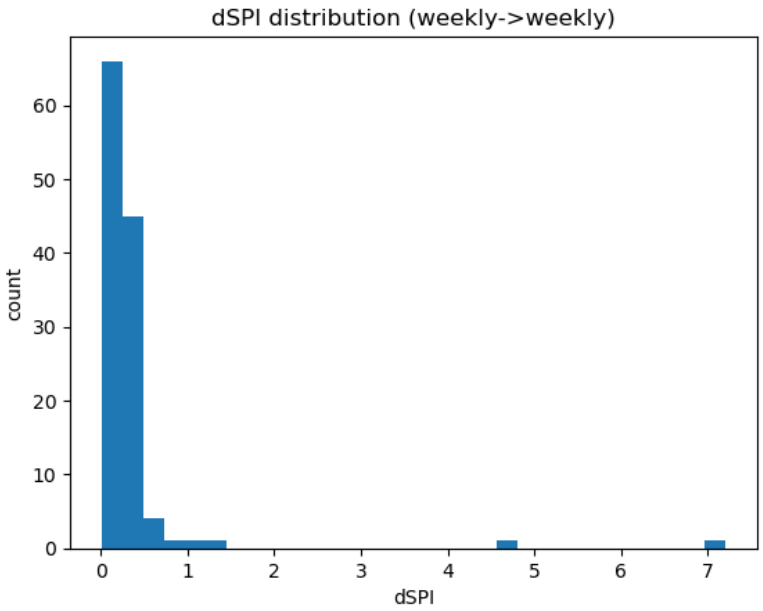

図 10: weekly→weekly イベントにおける ΔSPI の分布

表 7: $SPI_{\text{after}} \geq 1$ を満たすイベントの件数と ΔSPI 中央値

モード遷移	n（$SPI_{\text{after}} \geq 1$）	全体件数	ΔSPI 中央値
weekly→weekly	89	109	0.376
short→short	44	53	1.708
short→weekly	6	16	-0.800

表 8: モード遷移別の $SPI_{\text{after}} \geq 1$ 到達率

モード遷移	到達率（$SPI_{\text{after}} \geq 1$）
weekly→weekly	0.817
short→short	0.830
short→weekly	0.375

6.1.8 小括

weekly→weekly を主対象とした結果，ΔSPI は正の方向に有意に偏っており，実績入力後に SPI が改善方向へ変化する事例が多く観測された。short モードやモード遷移を含む結果は算出ルールの影響が残るため，補足的解釈に留める。

なお，ログインの有無でイベントを分けて確認したところ，ログイン後に入力された 118 件のイベントでは，$\Delta\tilde{\text{S}}\text{PI} = 0.442$，平均 1.22 となり，全体（中央値 0.371，平均 0.92）よりも改善幅が大きかった。表 9 に，ログイン有無別の指標を示す。ここでは `loginFlag`=1（当日にサインインしている）を「警告・指標を閲覧した可能性が高い」イベントとして抽出したサブセットであり，閲覧可能性の違いによる改善幅の差分を確認する位置づけである。

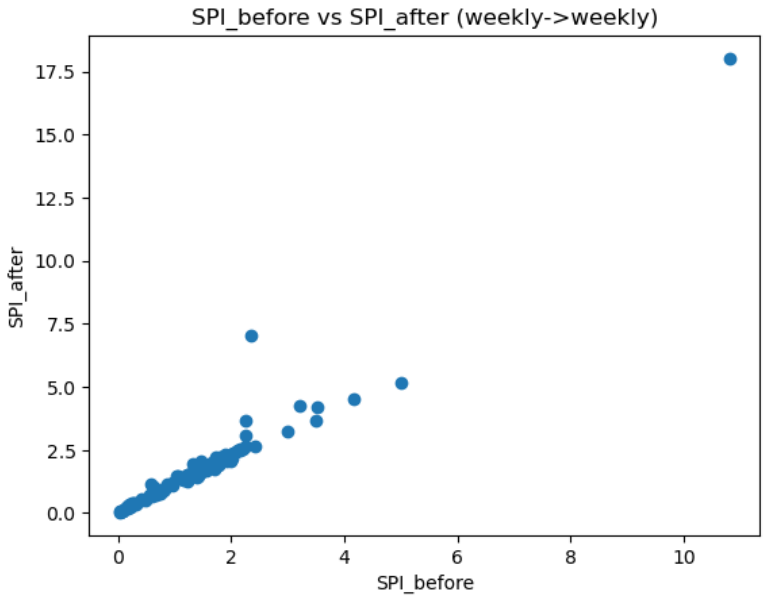

図 11: weekly→weekly イベントにおける SPI の入力前後比較

表 9: ログイン有無別の ΔSPI 集計（weekly モード中心）

ラベル	n	$\Delta\tilde{\mathrm{SPI}}$	平均 ΔSPI	$\tilde{\mathrm{SPI}}_{\mathrm{before}}$	$\tilde{\mathrm{SPI}}_{\mathrm{after}}$
全イベント	178	0.371	0.92	1.37	1.88
ログイン有り	118	0.442	1.22	1.07	1.79

6.2 分析 2：EAC に基づく将来予測と行動変化

6.2.1 遅延予測イベントの概要

本分析では，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抽出した。その結果，分析対象期間中において 27 件の遅延予測イベントが確認された。

各遅延予測イベントについて，イベント発生日の直前および直後における利用者の行動変化を評価するため，一定期間における実績作業時間を集計し，前後比較を行った。

6.2.2 遅延予測イベント前後の作業量変化

表 10 に，遅延予測イベント前後における 1 日あたり平均作業時間（7 日平均）の記述統計量を示す。

遅延予測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・中央値ともに減少が確認された。差分（after–before）の平均は -6.00 分，中央値は -5.0 分であり，73% のイベントで作業量が減少していた。

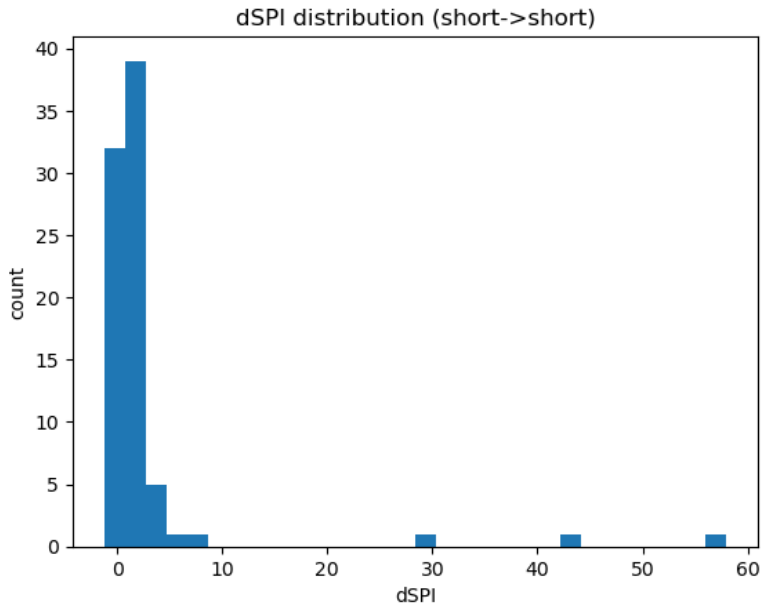

図 12: short→short イベントにおける ΔSPI の分布

表 10: 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

加えて，ログインが確認された 21 件のイベントに絞ると，イベント前後の平均作業時間は 31.93 分から 23.08 分へ低下し，平均差分は -8.84 分，中央値は -5.0 分であった。ログイン有無別の集計を表 11 に示す。

図 13 に，遅延予測イベント前後の平均作業時間の分布を示す。多くのイベントにおいて，イベント後の作業量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。

6.2.3　統計検定による前後差の検証

遅延予測イベント前後の作業時間差について，差分データの分布特性を確認するため，Shapiro–Wilk 検定および対応のある t 検定を試行したが，欠損を含むため統計量が算出されず（いずれも NaN），仮説検定は実施できなかった。記述統計のみからは，作業量が増加した割合は 0.27，減少が 0.73 であり，平均効果量（Cohen's d）は -0.13 と小さかった。

表 11: 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

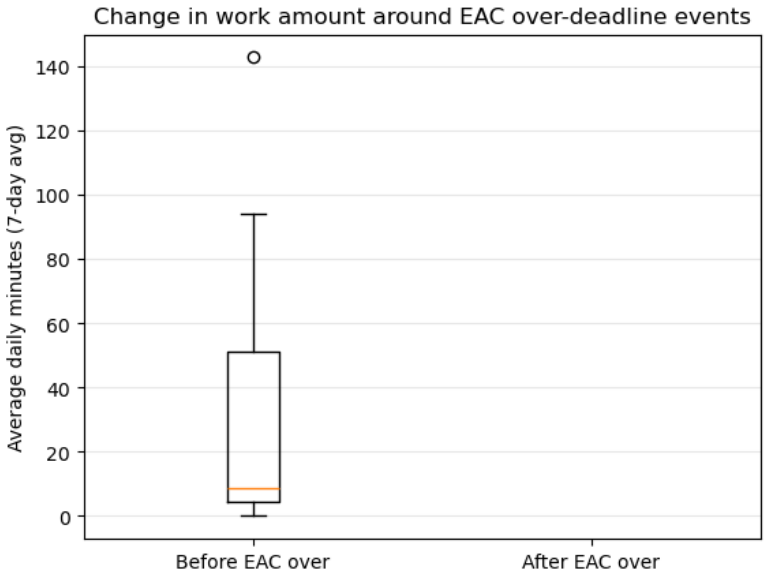

図 13: 遅延予測イベント前後の平均作業時間の分布

6.2.4 考察

以上の結果より，EAC によって将来的な締切超過が示唆された場合，多くの利用者において，直後の作業量が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 が利用者に対して即時的な行動修正を促す警告として機能したというよりも，将来的な進捗状態を再評価させる指標として作用した可能性を示唆する。すなわち，EAC による将来予測の提示が，自己調整学習における Self-evaluation（将来状態の評価）を喚起し，作業計画の見直しや一時的な行動抑制といった形で行動に反映されたと考えられる。

ただし，本分析では「締切超過が提示された」時刻を，システムが EAC を計算したイベント時刻として扱っており，利用者が実際に警告を閲覧したかどうかを示すログは取得していない。警告閲覧の有無を考慮した行動差分の検証は今後の課題である。

この点で，短期的な進捗評価に基づき行動修正（Self-control）を直接的に促す SPI と比較すると，EAC は中長期的な見通しの再評価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 分析の概要

本分析では，SPI や EAC に基づいて生成される「今日のプラン」が，利用者の実際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する。当日提示されたタスクと実行されたタス

クの対応関係，ならびに計画作業時間と実績作業時間の関係に着目し，行動提示と実行行動との整合性を評価する。

本分析では，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 と定義する。また，当日に実績作業時間が記録されたタスクを実行されたタスクとして扱う。

6.3.2 結果

まず，当日プランに含まれるタスクが実際に実行されやすいかを確認するため，planned task と unplanned task の実行率を算出した。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。一方で，unplanned task の実行率はこれより低く，当日プランに含まれるか否かが行動選択に影響していることが示された。

次に，実行が確認された日に着目し，実行タスクの内訳を分析した。実行タスクが存在した日は 13 日であり，これらの日に実行されたタスクは，すべて planned task 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。

さらに，計画作業時間と実績作業時間の関係を評価するため，planned task に対する時間遵守率（実績作業時間／計画作業時間）を算出した。その結果，時間遵守率には日ごとのばらつきが確認され，計画通りの時間配分が常に達成されるわけではないことが示された。

6.3.3 考察

以上の結果より，SPI や 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，planned task の実行率が約 34% であったことから，当日プランに含まれることが，タスクが実行される確率を高める要因の一つであると考えられる。

一方で，実行が確認された日については，実行されたタスクがすべて planned task であった。この結果は，いったん行動が選択される場合には，当日プランに基づいたタスクが優先的に実行される傾向があることを示唆している。ただし，本結果は限られたデータ数に基づくものであり，今後データ数の増加に伴い，異なる傾向が観測される可能性がある。

また，時間遵守率に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，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ると考えられる。このことは，本システムにおける「今日のプラン」が，行動を強制するものではなく，利用者の自己調整を前提とした行動指針として機能していることを示す結果と解釈できる。

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，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，遅延閾値の交差に限定せず，$\Delta\mathrm{SPI} \neq 0$ となった全入力イベントを対象に，日次実績入力を契機とした進捗状態の再認知と行動変化を検証した。特に尺度の一貫性が保たれる weekly→weekly 条件下において，ΔSPI は有意に正の方向へ偏っていた。この結果は，遅延状態

（SPI ≤ 1）の把握だけでなく，改善・悪化方向を含む進捗状態の更新自体が，実績入力とともに日常的な進捗認知を誘発し得ることを示唆する。すなわち，本システムは自己モニタリング（実績の記録）を起点として，進捗効率の変化を可視化することで，遅延・改善の双方を継続的に認知させる基盤を形成していると考えられる。

分析 2 では，EAC によって将来的な締切超過が示唆された場合に，直後の作業量が有意に減少する傾向が確認された。この結果は，EAC が即時的な行動量の増加を促す警告として機能したというよりも，将来状態の再評価を喚起し，計画の見直しや戦略の再構成といった中長期的な調整行動につながる余地を与えた可能性を示している。一方で，EAC の提示がタスクの達成不可能性を強く認知させ，作業意欲の低下やタスクの事実上の棚上げといった，**負の自己制御**につながった可能性も否定できない。本研究の定量分析は作業量の増減に基づく評価に限定されており，締切変更やタスク破棄といった調整行動の質までを追跡できていない点は，本研究の定量分析における限界点の一つである。

分析 3 では，SPI や 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とどの程度整合するかを検証した。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**計画されたタスク（planned task）**に集中する傾向が確認された。この結果は,「今日のプラン」が行動を強制するものではない一方で，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を示唆する。また，時間遵守率にばらつきが見られたことは，推奨作業時間が利用者のキャパシティ上限内で提示され，計画外タスクの実行を排除しないというシステム設計上の特徴と整合的であり，本機構が利用者の裁量を前提とした支援として機能していることを示している。

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。

図 14 に，本研究で提案した動的フィードバック枠組みの概念図を示す。本モデルでは，日次の実績入力を起点として進捗評価と行動提示が循環的に更新され，自己調整プロセスが継続的に支援される構造となっている。

すなわち，従来の ToDo 管理のように締切直前の気づきに依存するのではなく，日次の実績と指標更新を通じて進捗認知と将来評価を早期に生起させ，行動案へ接続する動的な仕組みが，個人タスクにおける新たな自己調整支援機構として機能し得ることが示唆された。

一方で，特に分析 2 で示唆されたように，EAC による将来リスクの認知をどのように具体的な行動修正へ接続するかは，本研究における最も重要な課題である。今後は，負の自己制御が生じる状況を詳細に特定した上で，将来予測の提示方法やフィードバック表現を改善し，過度な行動抑制や意欲低下を招かずに適切な行動修正を促す介入設計を検討する必要がある。また，より多様な利用者および長期利用データを用いた検証を通じて，行動選択と進捗指標の関係を精緻化することが，今後の課題として挙げられる。

6.5 脅威と限界（Threats to Validity）

- 内的妥当性：通知タイミングや学期行事，試験前後などの外部要因は統制していない。イベント間隔を 7 日以上とすることで独立性を高めたが，完全には排除できない。

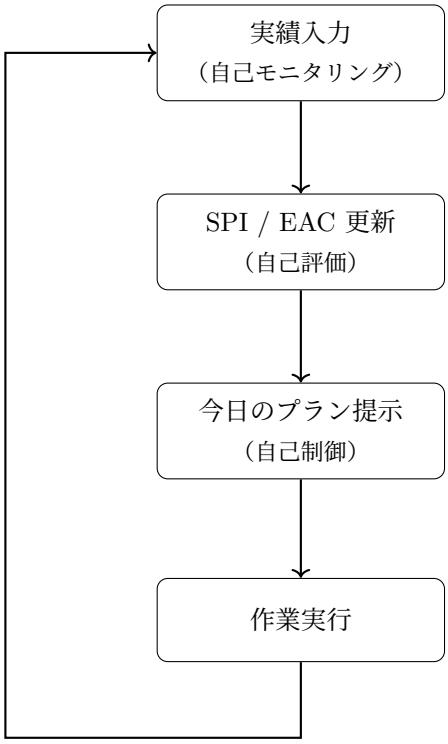

図 14: 日次実績ログに基づく自己調整プロセスを実装した動的フィードバック枠組み

- 外的妥当性：データは主に学習系タスクの大学生利用に偏っており，社会人や短期プロジェクトへの一般化は未検証である。
- 構成概念妥当性：遅延認知を SPI，締切リスク認知を EAC で近似しているが，主観的遅延感やストレスとの対応を測定していない。
- 結果解釈の限界：RQ3 は 1 ユーザ・9 件と極小であり，効果量と信頼区間のみの探索的報告とした。

7　まとめ

本研究では，個人の長期タスクにおいて生じやすい「遅延に気づくのが締切直前になる」「計画と実績の乖離を日常的に評価しにくい」という課題に対し，日次実績ログに基づいて進捗指標を継続的に再計算し，行動選択へ接続する個人タスク進捗管理モデルを提案し，その実装と実利用ログに基づく評価を行った。提案モデルは，実績ペースを表す pace7d，進捗効率を表す SPI，将来の完了見込みを表す EAC を日次で更新し，それらに基づいて「今日のプラン」を生成・提示することで，自己調整プロセス（自己モニタリング・自己評価・自己制御）の循環をシステム上で支援することを狙ったものである。

評価では，(1) 週間評価フェーズにおける SPI の改善効果，(2) EAC に基づく将来予測提示と行動変化，(3)「今日のプラン」提示と実行行動の整合性の三つの観点から分析を行った。その結果，分析 1 では，$\Delta\mathrm{SPI} \neq 0$ となった全入力イベントを通じて，特に weekly→weekly 条件で $\Delta\mathrm{SPI}$ が正方向に有意に偏り（$p < 0.001$），日次の実績更新が遅延状態の把握に限らず，進捗変化（改善・

悪化）の認知を伴う契機として機能し得ることが示された。分析 2 では，EAC が締切超過を示唆した局面において，直後の作業量が有意に減少する傾向が確認され，将来リスクの提示が即時的な作業量増加という形ではなく，中長期的な再評価や計画見直しを促す可能性が示唆された。分析 3 では,「今日のプラン」に含まれるタスクが実行されやすく，実行が確認された日には実行タスクが計画されたタスクに集中する傾向が確認された一方，作業時間は状況に応じて柔軟に調整されており，本機構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指針として作用していることが示された。

以上を総合すると，本研究で提案したモデルは，日次実績ログを起点として進捗効率の認知（SPI），将来リスクの評価（EAC），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おり，従来の ToDo 管理のように締切直前の気づきに依存するのではなく，日次の実績と指標更新を通じて進捗認知と将来評価を早期に生起させ，行動案へ接続する仕組みが，個人タスクにおける新たな自己調整支援機構として機能し得ることが示唆された。

一方で，本研究にはいくつかの限界がある。第一に，分析 2 で示唆された作業量減少の背景について，本研究の定量ログのみでは計画変更・タスク破棄・締切修正といった調整行動の質を追跡できず，負の自己制御（意欲低下や棚上げ）との区別が困難であった。第二に，分析 3 において計画外タスクの実行がほとんど観測されなかったなど，利用状況の多様性が十分とは言えず，一般化可能性の評価には追加データが必要である。

今後の課題として，EAC による将来リスクの認知をどのように具体的な行動修正へ接続するかが最も重要である。リスク提示が過度な行動抑制や意欲低下を招く条件を特定し，提示表現の改善や代替行動案の提示，計画再構成支援などの介入設計に結び付ける。併せて，多様な利用者・長期利用データを用いた検証と，主観的遅延感やモチベーションのアンケート併録，EAC 提示方法の A/B テスト，締切変更・タスク棚上げといった調整行動ログの収集を組み合わせ，行動変化の因果的理解と個人差に応じたフィードバック戦略の精緻化を進める。

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　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。

A.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。

A.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。

A.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。

A.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

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

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，

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

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

A.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。

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

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。

A.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。

A.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。

A.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
```