

令和元年度春期応用情報技術者試験（問 9）徹底攻略・復習詳細レポート：クラウド移行におけるスケジュール立案とクリティカルパス分析・短縮手法

1. プロジェクト背景と課題構造の定量的分析

オンプレミス環境から SaaS/PaaS を中心としたクラウド環境へのシステム移行プロジェクトにおいて、スケジュール管理と要件定義の整合性を維持することは、プロジェクトの成否を決定づける極めて重要な要素です。クラウドサービスの活用は、開発費用や運用コストの削減、事業継続性の向上といった多くの構造的メリットをもたらす一方で、サービス固有の制約や既存業務プロセスとのギャップを生じさせやすい側面を持っています。本ケーススタディは、限定されたリソースと厳格な納期制約（120 日以内）のもとで、いかにネットワーク図を駆使してクリティカルパスを特定し、状況の変化に応じた適切なスケジュール調整手法（クラッシングおよびファストトラッキング）を選択すべきかを学ぶ上で、プロジェクトマネジメント（PM）理論の実務応用における極めて重要な示唆を含んでいます。

プロジェクト概要と前提条件の整理

P 社におけるシステム更改プロジェクトの全体像、移行目的、採用するクラウドサービス、および開発体制についての基本情報は、下表の通り整理されます。

項目	詳細内容
対象企業（P 社）	家電製品の製造・販売を行う中堅企業
システム更改の目的	開発コストおよび運用費用の削減、事業継続性の確保
目標スケジュール	プロジェクト開始から 120 日以内のシステム更改完了
採用クラウドサービス	会計システム: R 社製造業向け SaaS 人事給与システム: R 社 PaaS（セキュリティが強化された新 OS・新ミドルウェア採用）
体制および担当業務	情報システム部: Q 部長（指示役）、S 課長（PM） 開発要員: 業務仕様チーム[2 名]、環境構築チーム[4 名]、設計チーム[12 名]、製造チーム[12 名]（※[]内は P 社の開発要員数） 外部ベンダー: R 社（各作業で協働）

モジュール構成	人事給与システム: 人事、給与、共通の 3 モジュールで構成
---------	--------------------------------

各システムの詳細な改修要件および構築手法は以下の通りです。

- **会計システム（SaaS）の改修要件:** 経営企画部および経理部からの財務データ取得・集計・分析機能に対する要求に基づき、P 社の経理業務に適合させるため R 社の SaaS に対して小規模なカスタマイズを実施し、設定・テストを行います。
- **人事給与システム（PaaS）の改修要件:** 総務人事部からの「P 社特有の人事制度及び職位に基づく給与体系への対応と、プログラムを改修することなく、人事評価区分及び給与区分の追加・削除ができること」という要求を満たすため、現行ソフトウェアを改修して R 社の PaaS 上に配置・実行します。改修作業は外部委託を行わず、P 社の情報システム部の開発要員のみで実施し、同等の規模・難易度を持つ「人事」「給与」「共通」の 3 モジュールを並行して開発する体制をとります。

「So What?」（分析から得られる示唆と課題構造の深掘り）

本プロジェクト構造の定量分析から得られる重要な洞察は、「PaaS 移行における 3 モジュール並行開発構造」と「他プロジェクトとのリソース共有」の組み合わせが、余剰バッファの存在しない脆弱なスケジュール構造を生み出しているという点です。

人事給与システムでは、設計チーム（12 名）および製造チーム（12 名）がそれぞれ 4 名ずつ 3 つのグループに分かれ、3 モジュールを同時に並行開発する計画となっています。一見すると効率的な並行作業に見えますが、各グループは担当モジュールの作業が完了した直後、「既に計画されている他のプロジェクトの作業」へ即座に移動する前提となっています。つまり、各工程間やリソースに時間的・人的バッファが全く組み込まれていません。初期の要件定義や前工程で僅かでも遅延が生じた場合、その遅延が後続工程へダイレクトに波及し、プロジェクト全体の目標である 120 日制限を即座に圧迫する極めてリスクの高い構造となっています。

初期スケジュールの構造とリスク要因を定量的に捉えたところで、続いてアローダイアグラム（作業工程図）に基づくネットワーク解析と計画変更基準の算出プロセスへ進みます。

2. 設問 1：作業工程図（ネットワーク図）の計算解法と計画見直し基準

アローダイアグラム（作業工程図）において、各作業の最早開始日（ES: Early Start）や最遅終了日（LF: Late Finish）を正確に算出することは、スケジュール上の余裕時間（スラック/フロート）を把握し、プロジェクトの遅延リスクを未然に制御するための基礎的かつ最重要な技術です。また、プロジェクト計画は固定的なものではなく、初期フェーズで判明した事実に基づいて動的に見直す「チェンジマネジメント（変更管理）」の基準をあらかじめ組み込んでおくことが、納期遵守と品質確保を両立させる上で不可欠となります。

設問 1 (1) 空欄[a]・[b]の導出プロセスの完全解説

図 1 の作業工程図における空欄[a]および[b]の数値は、先行作業からの順方向計算（フォワードパス）および後続作業からの逆方向計算（バックワードパス）の原理に基づいて論理的に導き出されます。

【会計システムの作業経路】

[開始] ——> A1 (10 日) ——> A2 (10 日) ——> [a] A3 (30 日) ——> C (10 日) ——> [終了]

【人事給与システムの作業経路（一部）】

[開始] ——> B1 (10 日) [LF: b] ——> B4 (36 日) [LF: 90] ——> B5 (20 日) ——> C (10 日) ——> [終了]

空欄[a]（作業 A3 の最早開始日 ES）の算出プロセス

フォワードパスでは、プロジェクト開始日を「1 日目」とし、先行作業の最早終了日（EF）に 1 日を加えることで、後続作業の最早開始日（ES）を求めます。

1. 作業 A1（フィット＆ギャップ分析）：所要日数は 10 日。開始日を 1 日目とすると、最早終了日（EF）は 10 日目（1 日目～10 日目）となります。
2. 作業 A2（カスタマイズ）：先行作業 A1 の完了翌日である 11 日目が最早開始日（ES）となります。所要日数は 10 日であるため、最早終了日（EF）は $10 + 10 = 20$ 日目（11 日目～20 日目）となります。
3. 作業 A3（SaaS の設定・テスト）：先行作業 A2 が 20 日目に完了するため、その翌日である $20 + 1 = 21$ 日目 が最早開始日（ES）となります。

- 正解: 21

空欄[b]（作業 B1 の最遅終了日 LF）の算出プロセス

バックワードパスでは、プロジェクト全体納期を遅らせないための限界期日を逆算します。後続作業の最遅開始日（LS）から 1 日を引いた前日が、先行作業の最遅終了日（LF）となります。

1. 作業 B4（プログラム製造・単体テスト）の最遅スケジュール解析: 図 1 より、作業 B4 の最遅終了日（LF）は「90」と与えられています。作業 B4 の所要日数は表 1 より 36 日です。作業 B4 を 90 日目までに完了させるための最遅開始日（LS）は、 $90 - 36 + 1 = 55$ 日目 となります。
2. 作業 B1（PaaS の設定・テスト）の最遅終了日（LF）の特定: 作業 B1 は作業 B4 の直前の先行作業です。作業 B4 が遅くとも 55 日目に開始されなければプロジェクト全体が遅延するため、先行する作業 B1 は必ず 54 日目の終わりまでに完了していなければなりません（ $55 - 1 = 54$ ）。

- 正解: 54

設問 1 (2) 作業一覧・工程図修正条件のアナリシス

Q 部長から指針として示された「フィット & ギャップ分析 (作業 A1) 完了時点での計画見直し」について、どのような状況で修正が必要になるかを分析します。

パッケージソフトウェアや SaaS を導入する際、業務要求と標準機能との適合度を検証する「フィット & ギャップ分析」は、事前想定 of 妥当性を確認する重要ゲートです。提案書の段階では「小規模のカスタマイズで対応可能」と評価されていても、詳細な仕様精査の結果、業務との乖離 (ギャップ) が想定以上に大きく、カスタマイズ工数や所要日数の増加を余儀なくされる場合があります。

- **修正が必要となる条件:** フィット & ギャップ分析の結果、カスタマイズの規模が事前の想定とかい離した場合 (想定以上のギャップが判明し、後続のカスタマイズ所要日数が増大する場合) に、作業一覧および作業工程図の修正が必要となります。

解答例 (IPA 模範解答) カスタマイズの規模が事前の想定とかい離した場合 (26 字 / 30 字以内)

「So What?」(戦略的意義の抽出と考察)

フィット & ギャップ分析 (A1) をプロジェクトの最初期 (1~10 日目) に配置し、その完了時点で計画再評価のゲート (関門) を設けることには、明確な PM 上の戦略的意義が存在します。

クラウド移行において最大の不確実性要素である「標準機能と固有業務の差異」を最序盤で確定させることで、万一カスタマイズ規模の膨張が判明した場合でも、クリティカルパス (人事給与側) に悪影響を及ぼす前に、SaaS 側の設定期間調整やベンダー (R 社) とのリソース再配分を早期に実施できます。このように、初期フェーズに品質・計画チェックポイントを配置するアーリーリスクマネジメントが設計されています。

初期計画の算定とリスク管理基準を整理したところで、次はプロジェクト全体のボトルネックであるクリティカルパスの特定と、納期短縮要求に対する代替案の定量比較解析へと進みます。

3. 設問 2: クリティカルパス特定と短縮手法 (クラッシング vs ファストトラッキング) の定量比較

プロジェクト進行中、事業上の要請により当初計画からの納期短縮 (スケジュール圧縮) を求められる場面は頻繁に発生します。スケジュール短縮を実施する際は、単に作業日数を削減するのではなく、クリティカルパス上の作業を対象とし、「直接コストの増加 (クラッシング)」と「手戻り・品質リスク (ファストトラッキング)」のトレードオフを定量的に評価した上で、経営的・運用的に最適な手法を選択することが求められます。

設問 2 (1)・(2) ボトルネック特定とファストトラッキングの構造

空欄 [c] (ボトルネック経路の特定)

まず、プロジェクト全体の所要日数を決定づけている経路を各ラインごとに計算・比較します。

- 会計システム経路 (A1～A3+C) : $10 + 10 + 30 + 10 = 60\text{日}$
- 人事給与システム (PaaS 環境構築経路 : B1+B4+B5+C) : $10 + 36 + 20 + 10 = 76\text{日}$
- 人事給与システム (開発経路 : B2+B3+B4+B5+C) : $30 + 24 + 36 + 20 + 10 = 120\text{日}$

最も所要日数が長く、プロジェクト全体の完了日 (120 日) を制約している経路は「B2-B3-B4-B5-C」です。この余裕時間 (フロート) がゼロである最長経路を「クリティカルパス」と呼びます。総務人事部からの開発期間短縮要請に応えるためには、このクリティカルパス上の作業 (特に B3～B5) を対象に短縮施策を講じる必要があります。

- 正解: クリティカルパス

空欄[d] (ファストトラッキングの対象作業と開始タイミングの算術ロジック)

案 2 では、要員を追加せずに先行作業と後続作業をオーバーラップさせる「ファストトラッキング」を採用します。

当初計画において、作業 B3 (ソフトウェア設計 : 所要 24 日) は 3 つのグループが「人事」「給与」「共通」の 3 モジュールを同時並行で設計していました。案 2 ではこれを変更し、12 名の設計チームを 1 つのグループに統合し、「人事」→「給与」→「共通」の順に順次設計を行います。1 モジュールあたりの設計期間は $24\text{日} \div 3 = 8\text{日}$ です。

1 つ目のモジュール (人事) の設計が完了するのは **8 日目** です。この 8 日目というタイミングは、作業 B3 全体が完了する 24 日目の **16 日前** ($24 - 8 = 16$) に相当します。すなわち、作業 B3 の全完了予定日の 16 日前 (人事モジュールの設計完了時点) において、完了したモジュールから直ちに後続作業に着手することが可能となります。したがって、並行して開始する後続作業は **B4** (プログラム製造・単体テスト) となります。

- 正解: B4

【案 2 : ファストトラッキングによるモジュール順次並行開発】

作業 B3 (設計) : [人事 (8 日)] [給与 (8 日)] [共通 (8 日)]

| <————— 16 日前 —————> |

作業 B4 (製造) : [人事 (12 日)] [給与 (12 日)] [共通 (12 日)]

設問 2 (3) 案 1 (クラッシング) と案 2 (ファストトラッキング) の定量評価・比較

S 課長が検討した「案 1 (要員追加＝クラッシング)」と「案 2 (順序変更＝ファストトラッキング)」の評価数値を定量的に導出します。

空欄[e] (案1の短縮日数の計算ロジック)

案1は、開発要員を追加投入して作業能力を高めるクラッシング案です。

1. 対象作業 (B3～B5) の当初総工数の算出:

- 対象作業の当初所要日数: 作業 B3 (24 日) + 作業 B4 (36 日) + 作業 B5 (20 日) = 80 日
- 当初体制: 12 名
- 総作業量 (工数): $80 \text{日} \times 12 \text{名} = 960 \text{人日}$

2. 追加人員の実効生産性の換算:

- 設計チーム・製造チームにそれぞれ 10 名を追加 (12 名→22 名体制)。
- 追加要員の生産性は業務要件理解のオーバーヘッドを考慮し「当初計画の 80%」と見込む。
- 追加 10 名分の実効開発能力: $10 \text{名} \times 80\% = 8 \text{名分}$

3. 変更後の体制における所要日数と短縮日数の算出:

- 実効合計開発能力: 当初 12 名 + 実効 8 名 = 20 名分
- 変更後の所要日数: $960 \text{人日} \div 20 \text{名} = 48 \text{日}$
- 短縮日数[e]: 当初 80 日 - 変更後 48 日 = 32 日

- 正解: 32

空欄[f] (案2の実質増加コストの計算ロジック)

案2は、他プロジェクトのリソースを優先配置することによる機会損失コストと、開発期間短縮による事業上のコスト削減効果を相殺して算出します。

1. 他プロジェクト遅延による増加コスト: 150 万円
2. 開発期間短縮 (16 日間) によるコスト削減効果: 開発期間短縮 1 日あたり 5 万円の事業運営コスト削減効果が見込まれるため: $16 \text{日} \times 5 \text{万円/日} = 80 \text{万円}$ の削減
3. 実質増加コスト [f] の算出: 実質増加コスト = 他プロジェクト遅延コスト (150 万円) - コスト削減効果 (80 万円) = 70 万円

- 正解: 70

両案の比較評価と経営判断の妥当性

評価指標	案 1（クラッシング）	案 2（ファストトラッキング）
短縮日数	32 日間（[e]の回答）	16 日間
増加コスト	320 万円（追加 20 名×32 日等の直接人件費）	70 万円（[f]の回答：実質増加分）

P 社および総務人事部が**案 2 を採用した経営上の妥当性**：案 1 は 32 日間の大幅な期間短縮が可能です
が、320 万円という高額な追加費用が発生します。一方、案 2 は短縮日数が 16 日間にとどまるものの、
実質増加コストを 70 万円（案 1 の約 4 分 1 以下）に抑えることができます。P 社は増加コストと短縮
ニーズのバランスを総合的に勘案し、投資対効果に優れた案 2 を選択したと言えます。

「So What?」（短縮手法の構造的リスクとトレードオフの深掘り）

クラッシング（案 1）とファストトラッキング（案 2）の本質的差異は、「**直接費用増大リスク**」と「**リ
ワーク（手戻り）リスク**」のトレードオフ構造にあります。

【短縮手法の構造的トレードオフ比較】

クラッシング（案 1：コスト投入型）

[資源追加] ——> コスト増（320 万円）+ ブルックスの法則リスク（コミュニケーション・習熟
遅れ）

ファストトラッキング（案 2：リスク受容型）

[作業重ね合わせ] ——> 低コスト（+70 万円）+ 設計未完了による大規模手戻り（リワーク）リス
ク

- **クラッシング（案 1）の特性**：新 OS・新ミドルウェア経験者を追加しても、固有の業務要件へ
の熟知には時間を要します（生産性 80%）。要員増加に伴うコミュニケーションラインの増大
（ブルックスの法則）や直接費用の激増が課題となります。
- **ファストトラッキング（案 2）の特性**：直接コストを最小限に抑えられる一方、先行する設計作
業（B3）が完全に完了しないまま後続の製造作業（B4）を並行稼働させるため、上流での仕様
変更や不備が下流の製造作業へ連鎖的に波及し、大規模な手戻り（リワーク）を惹起するリス
クを現場が受容することになります。

ファストトラッキングによる作業オーバーラップが惹起する手戻りリスクを抑え込むため、次章では品
質判定プロセス（品質ゲート）の再構築について論理的に総括します。

4. スケジュール短縮に伴う品質管理と実践的教訓の総括

ファストトラッキング（案2）を採用してスケジュールを短縮する場合、工程のオーバーラップによって生じる不具合の潜伏や手戻りの拡大を防ぐため、品質判定プロセス（品質ゲート）の細分化と最適化が絶対条件となります。

設問2（4）品質判定タイミングの見直しロジック

当初計画では、作業B3（ソフトウェア設計）および作業B4（プログラム製造・単体テスト）の品質判定は、「各作業項目とも全グループ（全3モジュール）の作業が完了した時点」でS課長が成果物の判定を行うルールとなっていました。

しかし案2では、3モジュールを「人事」→「給与」→「共通」の順に時期をずらして順次開発し、設計が完了したモジュールから直ちに製造へと進めます。もし品質判定タイミングを従来の「全モジュールの作業完了時」のまま据え置いた場合、最初に設計を終えて製造に入ったモジュールに設計不備が存在しても早期に発見できず、全モジュールの製造が進んだ段階で致命的な手戻りが発生してしまいます。

したがって、後続の製造作業へ不備を持ち込ませず手戻りを最小化するために、品質判定のタイミングを次のように見直す必要があります。

- **変更後の品質判定タイミング:** 各モジュールの各作業が完了した段階で即座に判定を行う体制に切り替えます。

解答例（IPA模範解答） 各モジュールの各作業が完了したタイミングで品質判定を行う。（33字 / 35字以内）

【品質判定プロセスの再構成（品質ゲートの細分化）】

（変更前：一括判定）

[全モジュール設計完了] —————> 【品質判定】 ———> [全モジュール製造開始]

（不備発見時の手戻りが甚大）

（変更後：段階的判定）

[人事設計完了] ———> 【品質判定】 ———> [人事製造開始]

[給与設計完了] —————> 【品質判定】 ———> [給与製造開始]

[共通設計完了] —————> 【品質判定】 ———> [共通製造開始]

(不備を早期封じ込め)

応用情報技術者試験における解法アプローチと実務への応用

本ケーススタディの解析から得られる、試験攻略および IT プロジェクトコンサルティング実務に直結する実践的教訓を 3 つの軸で整理します。

1. 計算問題における確実な解法手順（アローダイアグラム攻略）

- **ノードへの ES/EF/LS/LF の明記と数え上げ規則の標準化:** アローダイアグラムの各作業ノード（四角形）に対して、フォワードパス（ES, EF）とバックワードパス（LS, LF）の数値を漏れなく書き込みます。
- **実効生産性に基づく工数計算の標準化:** 要員追加時の生産性低下（例：80%）が指定されている場合、単なる人数加算ではなく「有効人数 = 追加人数 $\times$ 生産性」を算出し、総工数（人日）を有効人数で割り戻して所要日数を導出するステップを徹底します。

2. 記述式設問におけるキーワード抽出フレームワーク

- **「変更前」と「変更後」のギャップ構造の言語化:** 「どのような場合に」「どのように見直すか」という問いに対しては、問題文中の「当初の前提（全モジュール一括判定）」と「変更後の作業方式（モジュール順次着手）」の差分に着目します。「各モジュール」「各作業」「完了時」という必須キーワードを抽出し、指定文字数（30 字～35 字以内）の枠組みに合わせて緊密な論理を構成します。

3. 実務におけるクリティカルパス管理と品質保証の連動

- **スケジュール短縮と「品質ゲート（チェックポイント）」の同期設定:** ファストトラッキングを適用して作業をオーバーラップさせる際は、スケジュールの圧縮と引き換えに高まる手戻りリスクを抑え込むため、品質判定の単位をモジュール単位・工程単位へ細分化し、アーリーチェック（早期検証）の仕組みを同時構築することがプロジェクト成功の鍵となります。