Crossfields
Works Collaborated by Professionals

プロジェクトを成功に導くポイント

1. プロジェクト管理（進捗）前篇

株式会社クロスフィールド

庄司 堅太郎

クロスフィールド レポートページ TOP へ
http://www.crossfields.co.jp/reports/index.html

目次 （前篇）

はじめに

どのようなプロジェクトにおいても、あらかじめ計画された QCD（品質・コスト・納期）の達成が求められる。そのため、この３要素のマネジメントは非常に重要なものとなるが、外部からの変更・追加要素がなければ実際はそれぞれがトレードオフの関係にあるため、プロジェクトマネジャーは全体的なバランスを取りながらの非常に難しい舵取りを要求される。

そのために生み出されたマネジメント体系（PMBOK）は近年では相当洗練され、認知度も相当高いと思われるが、実際のところ重要となるのはその体系を具体的手法へと落とし込み、実行することであり、この点はマネジャーの方々は非常に苦労されている部分ではないかと思う。

本連載では「プロジェクトを成功に導くポイント」と称し、PMBOK 知識エリアごとに、当社での実績を基にプロジェクトマネジメント上のポイントをご紹介する。なお、マネジメント体系の基本は PMBOK であるため、その全体像や詳細な内容については、専門書を当たっていただければ幸いである。

第１回ではプロジェクト・タイムマネジメント知識エリアに含まれるプロセスの内、「監視・コントロールプロセス」に当たる【スケジュール・コントロール（＝進捗管理）】にフォーカスし、主にプロジェクトマネジャーの立場から、計画されたスケジュール・ベースライン（プロジェクト承認済の日程計画）をいかに遵守していくか、または変更が必要な場合いかに QCD を維持しながらスムーズにスケジュール・ベースラインを変更していくかを、即効性があり比較的効果の高いポイントに絞って説明する。前篇となる今回は「進捗の把握」を、後篇では「変更への対処」を主に紹介したい。

なお今回扱うプロジェクト事例の前提としては、IT システム導入プロジェクト（カスタムメイド、パッケージ問わず）を想定している。

進捗を正しく把握する

プロジェクトマネジャーからみると、日々の進捗把握や進捗会議における報告内容の正確性・信憑性がなければ、現在のプロジェクト状況がスケジュール・ベースラインからどれだけ乖離（＝超過達成／遅延）しているのか正確に判断できず、その結果として遅延に対して是正措置がタイムリーに取れない、または的外れな対応を取ってしまい効果が出ない、といった好ましくない事象が発生してしまう。よって、内容的にも時間的にも「進捗を正しく把握する」ということが、確実なスケジュール・コントロール実現の大前提である。

1 章 確実なコントロールのための仕掛け作り

タイム・マネジメントエリアの計画プロセス群（アクティビティ定義～資源・期間見積もり～スケジュール作成）については本文の対象外としているが、進捗を正しく把握するためには計画時点での仕掛け作りは不可欠であり、本章でそのチェックポイントを説明する。なおこれらのポイントは、計画時に盛り込まれていなくとも実行中でも確認及び改善が可能であるため、必要に応じて運用を見直していくことが肝要である。

■ スパンオブコントロール（管理範囲）の明確化

中規模以上のプロジェクトにおけるベンダサイドでの進捗会議では、例えば業務領域ごとのチームリードから、プロジェクトマネジャーに報告を行う、というようなコミュニケーションパスを通常取る。そのためにチームリードは事前に自身が管理すべき範囲、つまり自チームのタスク範囲についての進捗把握（必ずしも会議体での把握ではない）を行い、遅れているタスクについては原因分析やリカバリの見通し、及びチーム間タスクやマイルストーンへの影響を検討・判断した上での報告を行うべきである。
ただし、この一見普通と思われる進捗管理を完全に実現できているプロジェクトは経験上多くなく、チームリードの管理範囲が不明である場合、進捗管理上好ましくない状況が発生することがある。

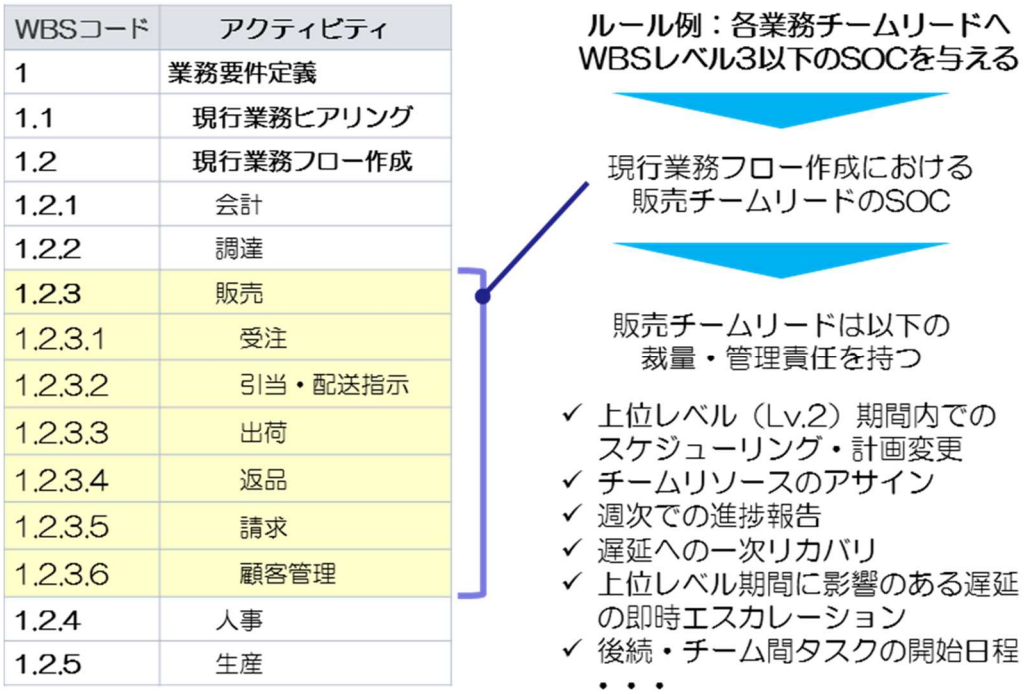
図 1：チームリードの管理範囲が不明瞭なために発生する事例

- 進捗会議の場で、チームタスクの遅れがリカバリ見通しもないまま報告される（チーム内部で十分に進捗確認・対策ができていない）
- スケジュール変更・組み替えをしてよい範囲がわからず、リソースの待ちや遊びが発生してしまっている
- 報告された遅延（リカバリ策の有無を問わず）が、他チームタスクの前提タスクとなっており、チーム外部や後続への意識がない
- マイルストーンを越える遅延が期限直前に報告される、など

これらに対するチェックポイントとしては、各進捗管理、報告の階層ごとに明確なスパンオブコントロール（以下、SOC）を設け、その範囲内での管理責任と裁量を与えることである。よく利用される実現方法としては、WBS 階層にそれぞれ責任者を割り当て、その単位での管理責任の割り当てと報告を行うのがよい。更にその責任者の SOC として、該

当 WBS 階層以下における進捗把握、人的リソース割り当てやアクティビティの組み替え、（コスト追加のない範囲での）遅延リカバリ実施といった責任と裁量を与える。

図 2：WBS 階層への責任者割り当てと SOC の主な例



一方で、各責任者の SOC を越える事象については、認識した時点で速やかにチーム間の調整（ヨコの調整）やエスカレーション（タテの調整）を開始することも、責任として与える必要がある。このように明確な SOC を与えることで、チームリード自身の責任と裁量がどこまであるのか、およびそれを越える事象が発生する可能性がある、または発生した時点でどのように行動し報告すべきなのかを意識付けすることができる。その結果、前述した事例のような丸投げエスカレーションや、重大遅延の報告遅れなどのリスクが低減され、プロジェクトマネジャーが管理すべき進捗状態の正確性、適時性が向上する。

また、これらの運用ルールはスケジュールマネジメント計画（プロジェクトマネジメント計画書の一要素）に盛り込み、確実にプロジェクト内に周知・実行徹底させることが必要である。

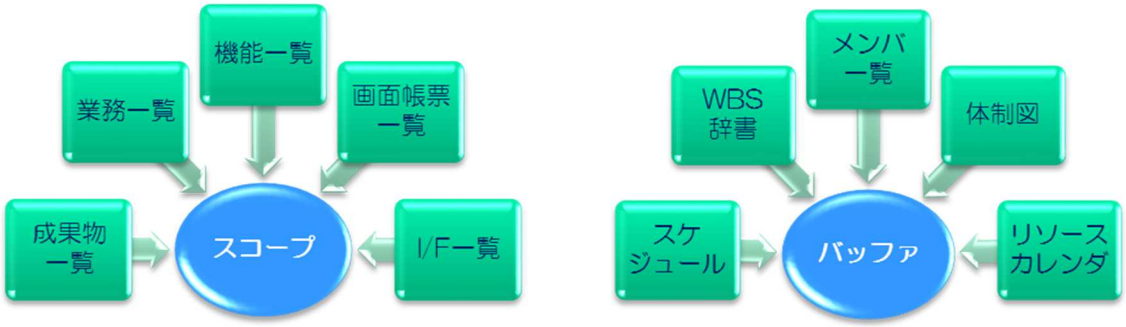
■ “一覧”でスコープとバッファを管理する

IT システム系のプロジェクトであれば、成果物一覧、業務一覧、機能一覧、画面一覧、インターフェース一覧など、様々な一覧表を作成していると思われるが、ここではタイム・マネジメントから見た必ず作成すべき一覧について、２つの観点で説明する。なおいずれの文書も他の知識エリア（スコープ、コスト、人的リソースマネジメント等）でもインプット・アウトプットとなるべき重要な文書であるが、あくまでもスケジュール・コントロールのための利用と限定して説明する。

1 つ目は「スコープを構成する一覧」である。これは、WBS（ワークパッケージ）の網羅性を確認するためのものであり、フェーズによって異なる場合がある。要件定義では業務一覧と機能一覧（業務一覧から機能一覧が派生し、機能単位で概要設計書を作成する、等）、開発では画面一覧・帳票一覧・インターフェース一覧、といった文書である。一度承認された WBS は、そのフェーズでの全アクティビティを表すもので、結果的にプロジェクト・スコープを表現しているものとなる。これらの“スコープを表現する”一覧群を整備し、プロジェクトメンバ（クライアント含む）と共有し都度確認し合う運用を取ることで、スコープ・クリープ（スコープの意図しない拡大）の未然防止やタスク網羅性の確認（特に統合テストやユーザー受入テスト）に有効となる。

二つ目は「バッファを管理するための一覧」である。ここで言う「バッファ」とは、時間的及び人的なものを指し、進捗管理上プロジェクトマネジャーが管理すべき最も重要な指標の一つである。具体的にはスケジュール表はもちろんのこと、WBS 辞書、プロジェクトメンバー覧（組織図）及びリソースカレンダー（要員計画表。チーム単位まで記載のあるもの）等である。特に人的リソースマネジメントについては、管理形式は種々あると思うが、ポイントはコストの発生する要員全員について、時系列にワークロード（作業工数）が表形式になっていること、である。これらの表を用い、スケジュール・コントロールの観点ではリソースバッファ（余裕リソース）の管理や作業時間超過リソース（＝ボトルネック）の把握、仕様変更時や遅延リカバリ時の要員の最適化・再配置・追加を行っていく。このようなバッファ管理やリソース競合の排除といった考えは、従来のスケジュールマネジメント手法（PERT 法やクリティカル・パス法）に比較してより人間心理や行動特性、組織的問題に配慮した手法であるクリティカル・チェーンマネジメント[1]の根幹を成しているものであり、近年のプロジェクトマネジメント上重要視されていることがわかる。なおバッファの考え方や管理ポイントについては、後篇でご紹介する。

図 3：スコープとバッファの管理を支える一覧表



多くのプロジェクト事例で、これらの一覧は計画段階で見積もりのために作成されるものの、実行時には更新がタイムリーではない、または存在が忘れ去られてしまい一覧としての意味を成さなくなっている。ここで紹介した一覧は、あくまでもスケジュール・コントロールからの必要性を説明したが、他の知識エリアからの要請含め、プロジェクトとして何を一覧として継続的に管理していくのか、計画段階で資料ごとのオーナー（管理責任者）や更新タイミングなど、運用まで決定しておくことが望ましい。

[1] エリヤフ・ゴールドラット、日本語訳 三本木亮
「クリティカルチェーン―なぜ、プロジェクトは予定どおりに進まないのか？」 ダイヤモンド社、2003 年 10 月

■ クライアント側でも進捗管理手法を検証する

一方で視点を変えて、クライアントサイドから見た場合、ベンダ側プロジェクトマネジャーから責任のある報告をもらえればよく、都度詳細なデータを確認するようなことはほとんどする必要がない。ただし、コントロールが実際に始まる前のスケジュールマネジメント計画段階ではベンダ側の進捗管理手法について、前述のようなコミュニケーション上のリスクを排除する仕組みになっているかどうかも含め、詳細に確認しておく必要がある。ここでよくある進捗管理上のブラックボックスが、ベンダ独自の進捗管理手法を採用する場合である。自社パッケージの独自の導入手法を確立している中小パッケージベンダに多いのだが、ベンダが「いつもやっており実績がある」や「我が社のパッケージ導入には最も適している」等の理由を挙げ、その手法を採用する。もちろんそれが最も効率的な場合もあると思うが、中にはウォーターフォール型のプロジェクトで見積もりにはガントチャートを利用するものの、実際のコントロールプロセスでは課題管理ツールのみを用いて、To-do も検討課題もあらゆるテストでのバグも全て課題として挙げてそれを順次対応する、という事例もあった。（当然、数千個の“課題”の前にプロジェクトはスタックした）

クライアント側ではベンダ側提案の管理手法を今回のプロジェクトに適用した場合も本当に効果的なのかを、ベンダ任せにせず、スケジュールマネジメント計画書とその確実な履行を詳細に検証すべきである。そのためには社内にスケジュールマネジメントに精通した要員を育成しておきプロジェクトに参加させたり（スポットでもよい）、必要であれば社外から専門家を調達したりすることも検討すべきである。また中長期的な取り組みとなるが、（他の知識エリアも同様）情報システム部門で複数のプロジェクト経験を通してプロジェクトマネジメントのナレッジを蓄積し、社内のマネジメント標準といった文書として形式知化していくことは非常に有効である。

２章 進捗の把握・報告

進捗の把握には、会議体の有無を問わず、なんらかの進捗報告が上位者に対して行われ、多段階のコミュニケーションパスが必要となる。そのため経験豊富なプロジェクトマネジャーでも、プロジェクトの規模が大きくなるにつれてタスク進捗の正確性を担保することが徐々に難しくなってしまう。正確な状態把握のために前章では計画段階でのポイントを説明したが、本章は報告と状態の把握段階でのポイントにフォーカスを当てて説明したい。

■ 進捗率基準（進捗メトリクス）を採用する

進捗報告を聞いていると、時折「進捗率 95%のまま２週間経過」のような報告が挙がることがある。２週連続で「あと少しで終わります」の報告を聞くことになるが、もちろん予定内であれば後続タスクへの影響はなく、一見問題はない（後篇にて説明する「パーキンソンの法則」のような好ましくない場合もあるが）のかも知れない。ただ多くの場合なにかがスタックしている、もしくは手が回っていないと見るべきであり、その理由を掘り起こす必要がある。このケースの問題は、「95%」という進捗率の基準を報告者（もしくは作業者）に任せている点であり、それを解消するには、①タスクをより細分化するか、②進捗率の基準をプロジェクトで決めておく、といった解決法が考えられる。①は成果物に対するワークパッケージという関係で作成されているタスクをさらに細分化することになり、管理コストは上昇してしまう（もちろん、レビュー記録表や資料修正版といった成果物が細分化されたプロセスごとに作成されるため、その単位での管理が望ましいこともある）。よって、ここでは②の進捗率基準（進捗メトリクス）を決める手法について説明する。

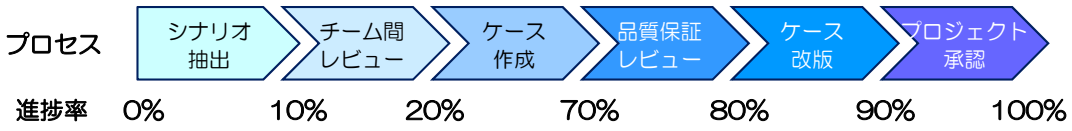
進捗メトリクスの決定基準としては、ボリューム及びプロセスの２つの分類がある。まずボリューム基準はページ数、ソースコード行数、機能（ファンクションポイント）数、テストケース数等といった「数・量」であり、その予定数値を母数として出来高の割合を進捗率とする。この基準は出来高の進捗自体が数値で表れるためわかりやすく、また進捗率の時間変化を捉えることで生産性も把握することができる一方、母数自体が予定数値であるため、着手後に膨らんでしまった場合（ソースコード行数が予定より多くなる等）の見直しが必要となることがある。よってこの基準は、成果物ボリュームが定量的に見積もりやすい成果物に対して利用するのがよい。例えば詳細設計書で機能数が規定されているコーディング等には、機能数を進捗率の母数とすることで（機能の規模が大きくバラついていない限り）比較的適用しやすいと考えられる。

次にプロセス基準は、「テストケース抽出⇒業務チームレビュー⇒テスト内容記述⇒品質チームレビュー⇒成果物改版⇒クライアント承認」のような一連の作業段階に対して、各作業が完了した段階での進捗率をあらかじめ決めておくものである。この基準は１つのタスク内部の作業段階を把握でき、母数も着手後の変更がないため利用しやすいが、成果物のボリュームとの関連がなく、さらに連続性のない飛び飛びの値を取ってしまうため、条件によっては１つの進捗率（＝作業段階）に長く留まってしまい本来の進捗率（この場合は１プロセス中の成果物の出来高）が見えづらくなる等の事象が発生することがある。これを避けるためにも、この基準は作成完了までに多段階のプロセスを経る成果物のうち各プロセスが長期にわたらないもの、に適用することが望ましいといえる。経験上ドキュメント作成には適用しやすいと考えているが、ヒアリングや課題解決を繰り返す要件定義フェーズよりも、事前に文書への要件が決まっている詳細設計書や単体・結合テスト仕様書等の方が作成プロセス自体の工数を事前に読みやすく、よりふさわしいと思われる。

最後に、２つの基準のハイブリッド型ももちろん考えられる。全体にはプロセス基準を適用するが、そのうち純粋な成果物作成の１プロセスのみにボリューム基準を適用する、等である。この方法は両者のデメリットを補完するかたちでより正確な基準となり得るが、あまり複雑な基準を設けることはかえって作業担当者やチームリードの管理コスト増加を招く危険性もあり、採用に当たってはアクティビティ自体の性質や成果物の重要度、及び遅延を許容できるスケジュールバッファ等の見合いで、どのレベルの進捗率精度を求めるのか（どれくらい“ガチガチ”にするのか）はプロジェクトマネジャーによって必ず判断が必要である。

図 ４：進捗率メトリクスの適用例

■ プロセス基準進捗率 適用例 （結合テスト仕様書作成タスクを想定）

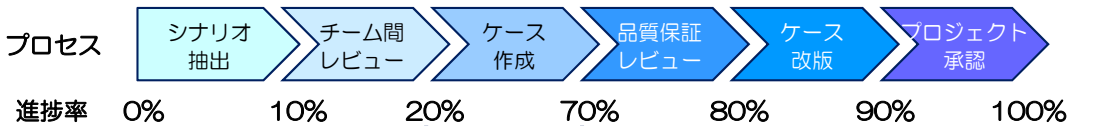


■ ボリューム基準進捗率 適用例 （コーディングタスクを想定）

$$\text{進捗率} = \frac{\text{コーディング済機能数}}{\text{見積もり機能数（FP 数）}} \quad \text{または} \quad \frac{\text{コーディング済コード行数}}{\text{見積もりコード行数}^{※}}$$

※ コード行数は見積もりがぶれやすいため、注意が必要

■ ハイブリッド基準進捗率 適用例



ケース作成プロセスの 50%幅のみボリューム基準を採用

$$\text{ケース作成プロセス時の進捗率} = \frac{\text{作成済ケース数}}{\text{総ケース数}} \times \frac{50^{※1}}{100} + 0.2^{※2}$$

※1：率の補正項　　※2：プロセススタート時の進捗率

なお、今回は進捗率のメトリクスについて紹介したが、プロジェクトマネジメント全般に適用可能な定量計測の枠組み及びモデル化手法を定義しているソフトウェア工学的手法にGQM[2]（Goal-Question-Metrics）がある。また経済産業省でも「ソフトウェアメトリクス高度化プロジェクト」を立ち上げ、定量的マネジメントの分野での研究成果を発表している[3]ので、興味のある方は参考にしていただきたい。

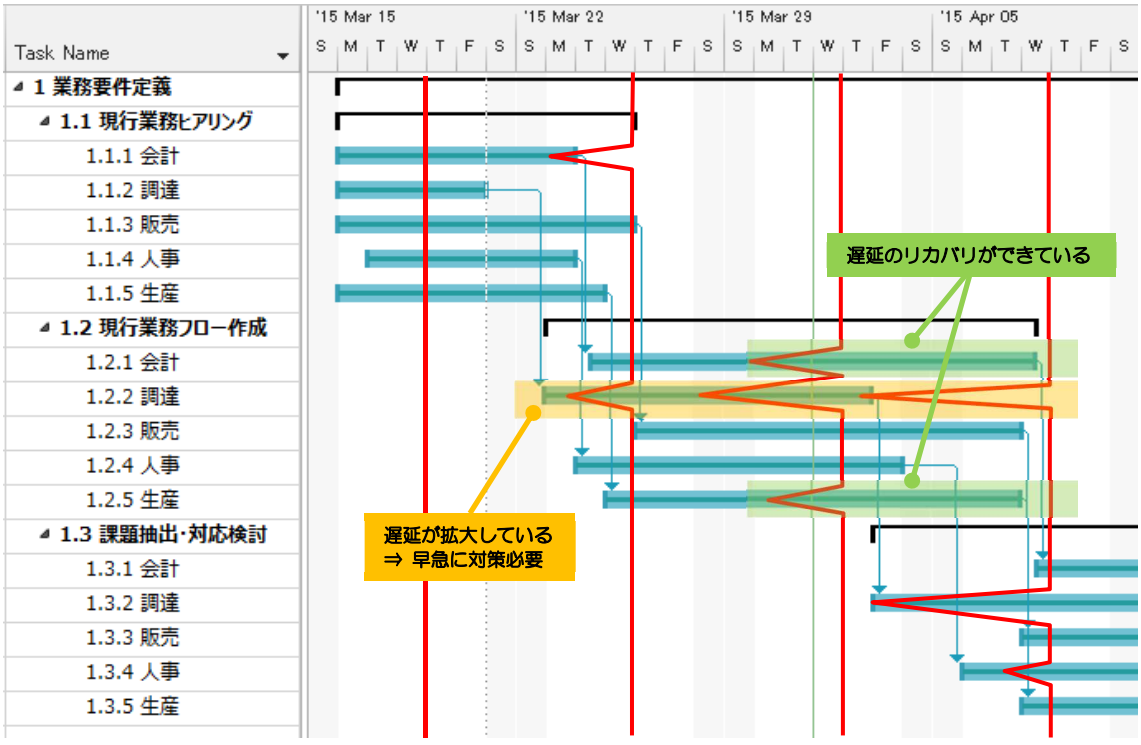
[2] Basili, V. R.: Software modeling and measurement: the Goal/Question/Metric paradigm, Sept. 1992.（日本語での解説本多数あり）
[3] http://www.meti.go.jp/policy/it_policy/softseibi/#p01_02

■ 時間的な状態遷移を観測する

進捗報告での内容、及び報告フォーマットの類は、プロジェクトマネジャーの経験やメンバのスキル、そのプロジェクトでの規定等により様々である。ただほとんどの内容が、「今週、どうだったか」や「今週の遅延は、2、3 日程度」といった現時点の報告を求めるものであり、特に文書での報告を行う場合にはこの程度が限界と思われる。ただしこれでは徐々に遅延幅を拡大しているタスクの発見や、学習曲線を見込んだリカバリ策（特に人員の追加）でもなかなか進捗が改善しない、等といった時間的な状態遷移を追うことは、そのままでは難しい。この点を簡便に表す手法として、ガントチャート上のイナズマ線を用いた報告をお勧めしたい。更に報告サイクル都度のイナズマ線は残しておいて、前回、前々回とイナズマ線の状態を比較できるようにしておけば、遅延拡大や解消の状態が一目で把握できる[4]ようになる。イナズマ線自体は、タスクごとに現在の進捗状態を表す方法として以前から利用されている手法だが、それを残して変遷を管理している事例は経験上多くない。

ただし実務上は、ガントチャート自体に複数のイナズマ線が残り続けることになるので進捗管理資料が見づらくなることも考えられ、タイミング限定での短期間の実施や、マネジャー側での把握資料として別途残しておく等の工夫が必要な場合もある。

図 5：過去のイナズマ線を残したガントチャート例

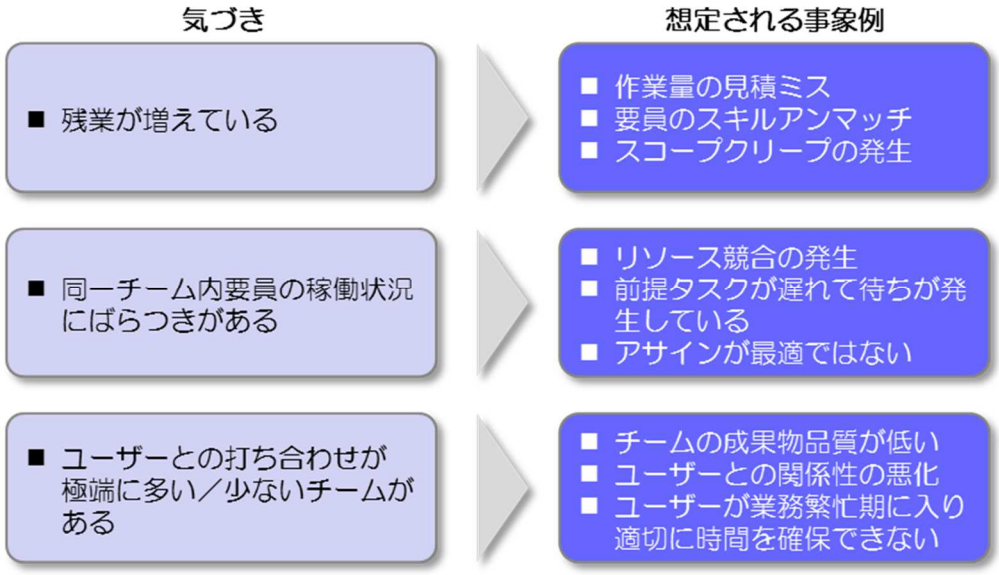


[4] 進捗管理ツール（Microsoft Project 等）を採用している場合、イナズマ線の作成や定期での自動作成、過去分の表示といった機能がすでに盛り込まれており、簡単に実現できる。

■ オフライン・コミュニケーションを活用する

ここまで正確な進捗を把握するための仕掛け作りや見える化・定量化について重きを置いてポイントを説明してきたが、会議・文書以外での“オフライン”の状態把握もマネジャーの役割として非常に重要であり、具体的には「現場を見る」ことと「対面で話す」ことが必要である。例えば必要なタイミングでプロジェクトルームに張り付いてメンバの動きを確認しておき、なんらかの“気づき”があればスケジュール計画上のミスや遅延の黄色信号の可能性を疑い、ざっくばらんにチームリードや担当メンバとコミュニケーションを取って状況を確認してみることで、遅延の発生を未然に防いだり、早期発見に繋がると考えている。地味な方法ではあるが、ハイパフォーマーと呼ばれるプロジェクトマネジャーは必ず実践している。

図 6：オフラインでの気づきと想定される事象例



ただしコミュニケーションを取る場合には、各チームリードの主体性や SOC を侵すような聞き方をしないように留意する必要がある。あくまでも一次的な進捗責任をチームリードに置いている場合は、リードを飛び越して担当者に直接根掘り葉掘り進捗状態を確認すべきではない。この辺の“オフライン”の感覚はマネジャーによって様々であるし、自由にコミュニケーションを取れるプロジェクトの雰囲気も重要であるが、あくまでも正式な状況確認は正規のコミュニケーションパスに則り行うのは当然である。またエスカレーションの場合も同じで、担当からプロジェクトマネジャーへ直接懸案等を上げるのではなく、まず担当にチームリードに上げるよう促すことで、チーム内で一度検討するように仕向けるべきである。

次回後篇では、クリティカルチェーン法の考え方を中心に「遅延への対処」を取り上げる。対処時に必要となるタスク間制約や行動心理の前提知識をまず説明し、その後具体的な遅延対処のチェックポイントについて解説したい。