はじめに

2004年10月にソフトウェア・エンジニアリング・センター（SEC）が発足して以来、SECは高品質のソフトウェアを効率よく開発・保守するための技術（ソフトウェア・エンジニアリング）の開発・普及を行っています。2005年4月には、エンタプライズ系活動の成果として、

『経営者が参画する要求品質の確保』（開発プロセス共有化部会）

『ITユーザとベンダのための定量的見積りの勧め』（見積り手法部会）

『ソフトウェア開発データ白書2005』（定量データ分析部会）

を発刊しました。これらは、プロジェクトを成功へ導くためのベストプラクティス、プロジェクトを失敗させないための予防策や手法を中心にまとめました。

しかし、残念ながら、証券取引所の障害をはじめ、社会インフラを担う情報システムの障害は、我が国の経済活動に大きな影響を及ぼしています。さらにIT業界においては、コストオーバーラン・プロジェクトの発生により、企業の経営を揺るがしかねない事態が発生しています。

これら憂慮すべき事態への対応として、我々は、予防策の重要性を認識しつつも、『憂慮すべき事態の兆候があれば早期に発見し、迅速かつ適切に対処することを推進して、社会・ユーザーへ貢献する必要がある』との認識に至りました。

そのための手法として、我々はITプロジェクトの『見える化』に着目しました。見える化は日本企業の強さの源泉であり、現場で起きている様々な事象をさらけ出し、問題を見えるようにすることがすべての出発点です。我々は、ITプロジェクトにおいても、見える化を実現したいと考えています。本書により、“強いソフトウェア開発現場”が作られ、その結果として情報システムの品質が向上することを願ってやみません。

2006年5月

プロジェクト見える化部会一同

ITプロジェクトの「見える化」 下流工程編

第1章

プロジェクトの「見える化」の概要

1.1 ITプロジェクトの実情

情報システムが人手のかかる単純処理を自動化するだけだった時代はとうに過ぎ去った。いまや情報システムは社会に深く入り込み、我々の生活になくてはならない存在になっている。情報システムやインターネットを使って直接営業するビジネスが普及するなど、情報システムを高度に活用できるかどうかは、経営者にとって生き残りを左右する非常に重要な要素になっている。

これらの情報システム開発で要求される機能は複雑かつ高度になり、要求される品質レベルも非常に高い。しかも、市場の激しい変化の中で競争しているため，情報システムの開発期間に対する短縮圧力は一層強くなっている。オープンシステムの影響などにより、引き続きシステム開発金額の値下げ圧力も強い。

下流工程で品質不良が頻発する

一方，ITプロジェクトの開発現場を見てみると、多様化した社会ニーズに応える機能がどのようなものかが、ユーザー（顧客）にとっても分かりにくくなってきている。「仕様決定」が難しくなり、ITプロジェクトの開発現場では仕様決定の遅れが多発している。また、システムの稼働環境や開発環境のオープン化に伴い、システム基盤のアーキテクチャ設計も、従来に比べて難度を増している。

しかし、開発期間の短縮圧力が強いため、ITプロジェクトがユーザーに納期の延期を言い出せず、下流工程が“突貫作業”になることが多い。場合によっては安定しない品質のまま納品することになり、本番稼働後にシステム障害を起こしてしまうこともある。これは、企業の社会的信用を大きく揺るがし、機会損失を発生させる。

1.2 ITプロジェクトの課題

短期間で高品質な情報システムを開発するためには様々な課題があるが、その

中でも「目に見えないソフトウェアを開発する」という点が、システム開発固有の難しさと言えるだろう。ごく単純化して言うと、ITプロジェクトには物の生産プロセスと異なる次のような特徴がある。

- 要件が目に見えない
- 開発プロセスそのものが目に見えないし、進捗状況も目に見えない
- 成果物の論理構成の正当性が目に見えない

こうした課題に対して今までにも様々な取り組みはあったが、「短期間で高品質な情報システムを作る」という昨今の状況の中では、まだ不十分である。この「見えないもの、分からないものを，いかに見えるようにするかの工夫」が、これからのITプロジェクトの成功を左右する重要な鍵である。

1.3 ITプロジェクトの「見える化」の目標

本書は、多くの問題を抱えているITプロジェクトのプロジェクト・マネージャ、品質管理担当者、プロジェクト・リーダーのために、プロジェクトで発生した事象、あるいは起こりつつある何かの予兆を「見える化」すること、さらに見えたものから真の問題が何かを「言える化」し、問題の対応策を策定する「直せる化」を含めて、現場に立脚した方法論を創出することを目標にしている。

いわゆる「見える化」は、単に「いかに見えるようにするか」のみならず、「どのような問題か特定し」、「いかに解決するか」という問題解決まで含んだものである。本書では、それぞれの切り口について、上記のとおり、狭義の「見える化」、「言える化」、「直せる化」の3つのキーワードを設定し、具体的な方法を示している。「言える化」、「直せる化」は本書で新たに提案する呼び方である。

「見える化」の対象はソフトウェア開発プロジェクトのライフサイクル全般で、非常に広範囲にわたる対策が求められる。その認識のもと、本書はまず「下流工程」に焦点を当ててITプロジェクトの「見える化」、「言える化」、「直せる化」に関する方法論を解説する。ここで言う下流工程とは**図表1-1**に示すように、プログラムの結合テストから総合テストまでの工程を指すものとする。

下流工程に焦点を当てる理由は、現実的な問題として、下流工程で多くの問題が一気に顕在化しがちであるものの、対応の時間や手段が限定されており、現場で解決への期待が非常に高い

ためである。また、下流工程では「見える化」の対象となる課題や対策が非常に具体的である。企業が初めて「見える化」に取り組む場合に、スムーズに導入できるという点も考慮した。

ITプロジェクトの「見える化」、「言える化」、「直せる化」に関する主なポイントは次の通りである。

「見える化」

①プロジェクト・マネジメントでは「分からないもの」の「見える化」が必要である。そのとき、より的確に把握するため、高い所から見るための「俯瞰図」を作成する。

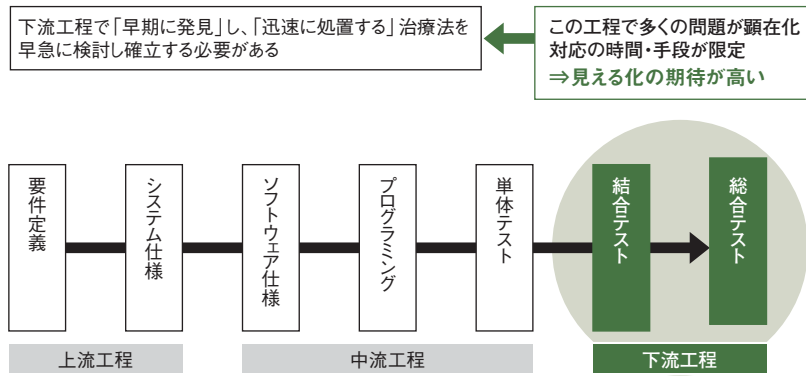
②開発中のプロジェクトに問題の兆候が現れているかどうかを自己診察するため、「チェックシート（自己評価シート）」を利用する。

③開発中のプロジェクトに問題の兆候が現れているかどうかを専門家がヒアリング診断するため、「チェックシート（ヒアリングシート）」を利用する。さらにヒアリングの際は、エビデンスとして確認するために用いる資料を明確にする。

④開発中のプロジェクトに問題の兆候が現れているかどうかを定量的に診断するため、代表的なツールを用いて測定データを取得する。それを実際のプロジェクトに適用して、定量的に問題を把握する。

⑤それぞれの「見える化」手法によって、すぐに問題が明らかになる場合には、その対応策をガイドラインとして提供する。

図表1-1●ITプロジェクトの下流工程で多くの問題が顕在化



※ ウォーターフォール型の開発工程例

「言える化」

複合的な問題の場合には、単独の「見える化」手法によってだけでは問題を特定できないことがある。その場合に、何が本当の問題かと「言える」ように、見える化した事象に基づいて原因を分析する必要がある。

①問題が何かを把握するため、「症例分類表群」を利用する。症例分類表群は、過去の失敗プロジェクトで生じた問題（ここでは「症例」と呼ぶ）をパターン化したものである。表の縦軸にプロジェクトの「何が（何を）」を、横軸にその症状「どうなっている（どうした）」をとり、各症例に関係する原因を多角的に分析できるようになっている。

②症例分類表群は「一般マップ」、「ヒアリングマップ」、「測定項目マップ」、「事例マップ」の4つで構成する。これらを用いて複合的な問題に対する統合的アプローチ手法を採る。

③統合的アプローチ手法を用いて、プロジェクトで生じている事象から原因を特定する。

「直せる化」

「見える化」や「言える化」によって問題の原因が明らかになったときの改善策の策定方法を検討する。改善策はプロジェクトの優先順位によって、またベンダー企業の文化によっても異なるので、本書で検討する領域は改善案を策定するときの方法論とする。

方法論の効果

ここで提示した手法は、チェックシートを除けば新しい手法ばかりだが、次のような大きな効果を確認した。是非、この方法論の導入を検討してほしい。

①「見える化」、「言える化」、「直せる化」のそれぞれの研究活動において、有効な手法を提示できた。

②「見える化」手法における改善策では、経験者の暗黙知を形式知に変えることができた。

③複合的な問題を特定するための方法として、ITプロジェクトにおける管理対象を定義し、それらと各手法との関連を整備できた。

本書で解説した「見える化」のためのツール（チェックシートや症例分類表群など）は、すべて付録資料として掲載した。その電子ファイル版（Excelファイル）もSECのホームページ（http://sec.ipa.go.jp/）からダウンロードできる。

SECは今後、この成果をベースに、上流工程や中流工程に遡ってプロジェクトを成功に導くための見える化手法を研究する。

第2章

プロジェクトの見える化マネジメント

ITプロジェクトのマネジメントの本質は、「見えないもの、分からないもの」をマネジメントすることにある。「見えないもの、分からないもの」には、あいまい性、不確実性に加えて、人がシステム開発の主役であることから生じる「問題の潜伏」や「問題の隠ぺい」も含まれる。マネジメントの質の向上は、これらを見える化することによってしか始められない。

2.1 見える化マネジメントとは

2.1.1 変革を迫られているITプロジェクトのマネジメント

ITプロジェクトのマネジメントの質を高めるために、プロジェクトは「健全なソフトウェア・エンジニアリング文化」を持つ組織への脱皮を迫られている。「健全なソフトウェア・エンジニアリング文化」としてどのような姿が望ましいかを考え、その特徴を示したのが**図表2-1**である。

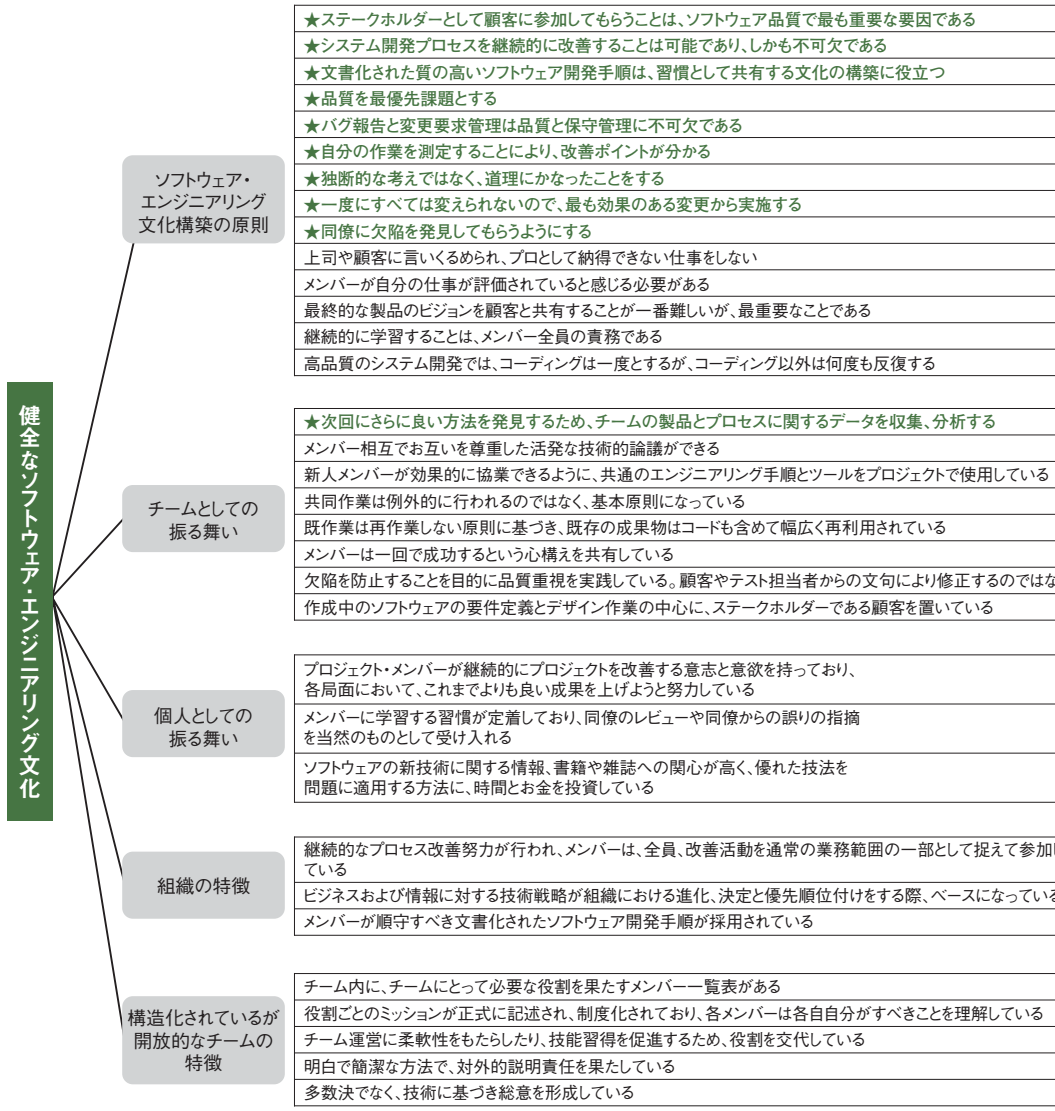
図中の「ソフトウェア・エンジニアリング文化構築の原則」には、より良いソフトウェア・エンジニアリング文化を醸成していく上で重要な原則を挙げている。「チームとしての振る舞い」では、システム構築において特に重要となるチームワークのあるべき特徴を整理している。「個人としての振る舞い」では、システム構築に携わる個人として、どのようにプロジェクトに参画するべきか、そのあり方を示している。

図中で★印の付いている項目には、本書で解説する「見える化」の手法が有効である。★印の大半は「ソフトウェア・エンジニアリング文化構築の原則」に付いており、「見える化」が「健全なソフトウェア・エンジニアリング文化」の構築に重要なだけでなく、ITプロジェクト全体で取り組むべき課題であることが分かる。

2.1.2 見える化マネジメント導入のポイント

「見えないもの、分からないもの」を

図表2-1●健全なソフトウェア・エンジニアリング文化の構築



★…見える化に関連する項目

マネジメントする方法論として「見える化」が不可欠なことは論を待たないが、漠然と「見る量」を増やせばよいというものではない。見るときのポイントを整理しておかないと、逆に忙しくなるだけで何の効果も期待できない。より的確に見てマネジメントにつなげるためには、「何をどのように見るか」の事前準備が必要になる。

また、経験豊富なプロジェクト・マネージャは、プロジェクトが始まると、すぐに俯瞰図の作成に取りかかる。俯瞰図は、高い所から見ることによって、それぞれの個別事情に埋没して見えにくい「プロジェクトの成否を左右する支配的な要因」（ドミナント・アイテムと呼ぶ）を把握するためのツールである。パレートの法則によると、プロジェクトの成否の8割を決めるのは2割のドミナント・アイテムである。これによって「木を見て、森を見ず」や「枝葉末節にこだわること」を避けられる。

プロジェクト・マネジメントとは、計画段階でドミナント・アイテムを知り、リスク要因の評価によって危険度を測りつつ、リスクを小さくする営みである。

2.1.3 見える化マネジメント導入の効果

見える化マネジメントは、**図表2-2**に示すように、プロジェクト・マネージャ自身が問題を早期に発見し、解決できるようにする効果がある。複数プロジェクトを横断的に支援し、プロジェクト・マネージャを指導する機能としてのPMO（Project Management Office）を運営する際にも、客観的に見て良い方向に導くために役立つ。

さらに、見える化マネジメントによって、開発ベンダーの経営層もタイムリにプロジェクトの状況を把握でき、経営の効率化を一層推進することができる。このようにプロジェクトの「問題」を内外からよく見えるようにすることが、課題を組織的に解決し、プロジェクトを成功に導くことにつながる。

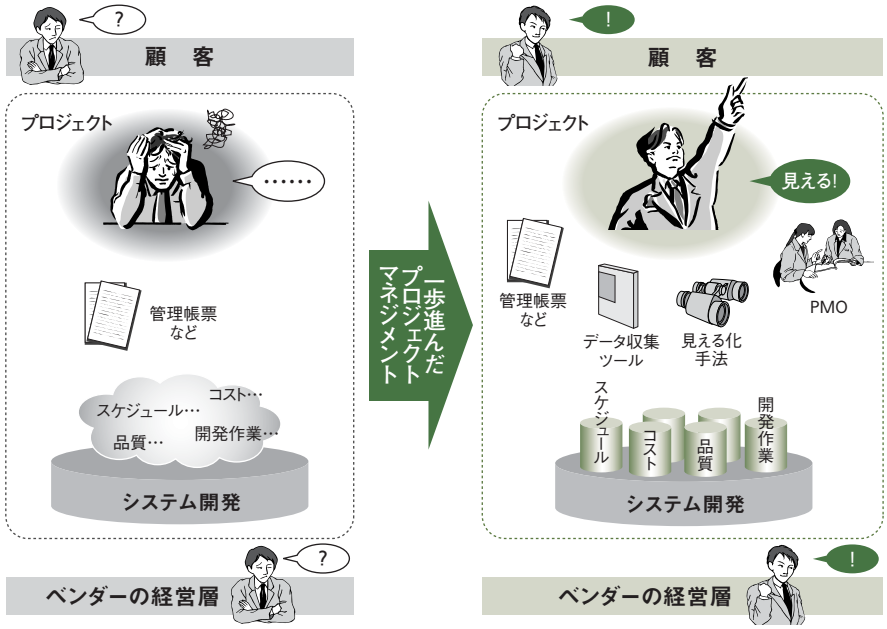
2.2 プロジェクトの“見える化”の全体フレーム

2.2.1 プロジェクト見える化の流れ

プロジェクトの見える化は、ITプロジェクトにおける問題点の早期発見を目的としており、問題の発見から改善に至る流れは、医療になぞらえて考えると分かりやすい。

医療では病気を見つけて治すまでの流れとして**図表2-3**のような流れがある。まず自分自身の体の異常を感じたときに、「自覚症状を把握する」というプロセスがある（自己診察）。例えば風

図表2-2●見える化マネジメント導入の効果



邪や疲労のような軽い症状だと判断した場合には、自己治療を施す場合がある。

また、定期健康診断により体を検査し、その結果に異常が認められる場合もある（健康診断）。自己診察や健康診断によって、その異常がどういうものなのかを調査するために、精密検査を行う（精密検査）。例えば関節に痛みがある場合、レントゲンを撮ってみて問題がないかをチェックする。

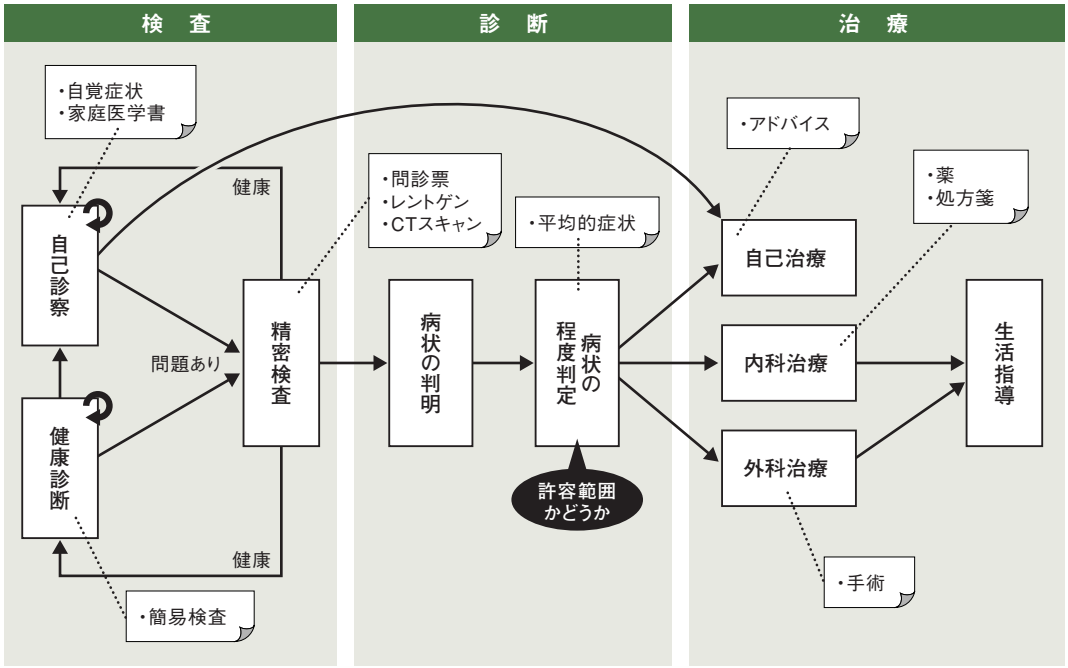
次に医師は、検査結果を基にどのような病状かを判断し、病名を決定する。さらに病状の程度を判定して、どのような治療を行うかを検討して実施するという流れになる。

治療方法にはいくつかの対処方法があるが、例えばここでは3つの実施レベルを想定し、アドバイスを中心とした「自己治療」、薬の服用等による「内科治療」、問題部位の切除などの「外科治療」に分類できる。

プロジェクトにおける問題発見から改善への流れも、**図表2-4**に示すよう

図表2-3●医療での検査・診断・治療のフロー

医療の現場で考えると…

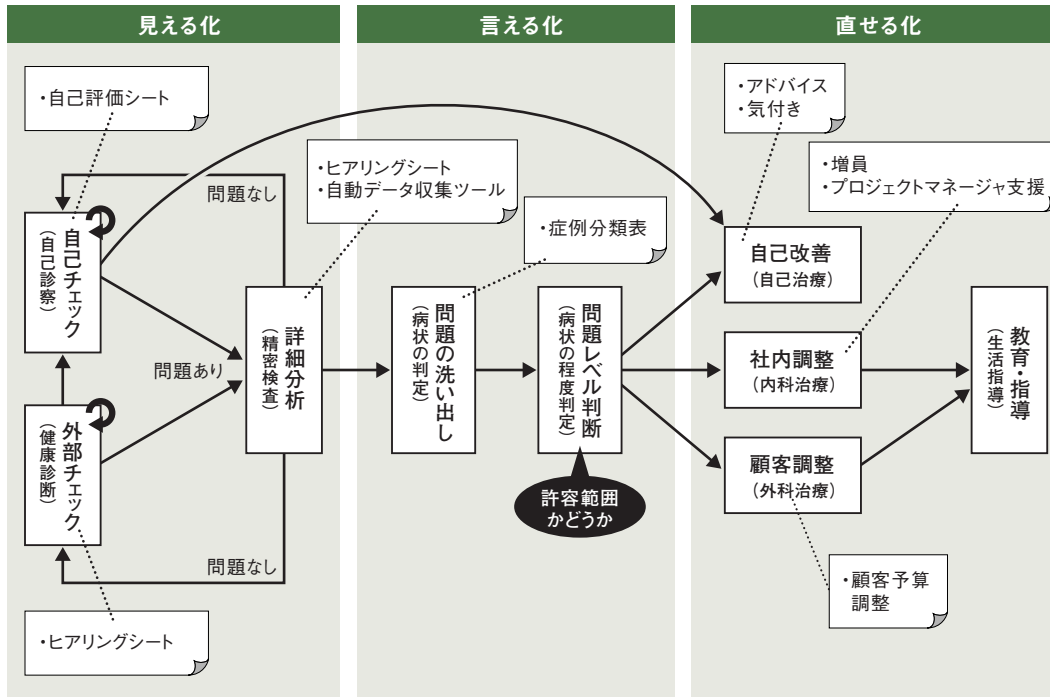


に、医療における「検査、診断、治療」を、「見える化、言える化、直せる化」として対比して考えることができる。

プロジェクトの異常を見つけて直すために、まず検査をしなければならない。プロジェクトそのものは人ではないので自覚症状というものはないが、プロジェクトに参画するメンバーやプロジェクト・マネージャ自身が、何らかの違和感をおぼえることはある。そういうとき、進捗管理表や課題管理表、プロジェクト・メンバーからの進捗報告などを基に、プロジェクト・マネージャがプロジェクトの状況を自己診察する(自己チェック)。プロジェクトの状況を確認するためには、SECが開発したチェックリスト（自己評価シート）などを利用できる。

PMOなどのプロジェクト外部の評価機関により、定期的にヒアリング診断で異常を見つけることもある（外部チェック)。後述する「ヒアリングシート」

図表2-4●プロジェクトの見える化・言える化・直せる化のフロー



などを利用できる。

こうした調査の結果を分析することで問題を洗い出し、プロジェクトの問題とそのレベルを明らかにする。そして、プロジェクトに対し問題とレベルに応じた種々の改善を実施するという流れになる。

改善には、プロジェクト内で実施できるものもあれば（自己改善）、プロジェクト外からの支援が必要で社内調整を通じて実施するもの（社内調整）、顧客との調整を経て提供機能の見直しやスケジュールの変更などを行うもの（顧客調整）などがある。

2.2.2 見える化手法

医療での検査に対応する「見える化」には、それぞれのプロセスに応じた手法を考えた。それぞれの手法と対応するツールを**図表2-5**に示す。それぞれの手法とツールの詳細は【第3章 見える化】で説明するが、ここでは概要を示す。

まずプロジェクトの見える化を実施する際に、最初に取りかかるべきことは「プロジェクト全体の把握」である。そのためにまず「俯瞰図の作成」を行うことが重要である。本書では「システム構成俯瞰図」、「ステークホルダー俯瞰図」、「スケジュール俯瞰図」、「要員遷移俯瞰図」の4つについて説明する。これらの俯瞰図を作成することによって、開発現場の全体を把握するとともに、改善活動を行う上で押さえるべき主要なポイントの把握に役立つ。

また、プロジェクトの様子をプロジェクト・マネージャから聞き出すための手法・ツールとして、チェックシートを使った「自己診察」、「ヒアリング診断」がある。チェックシートを提供することで、確認ポイントの漏れを防げる。併せて、「どのような項目について気をつけるべきか」というプロジェクト・マネージャの「気付き」が得られる効果が期待される。

「事例ベース診断」は、過去の失敗プロジェクトの事例データベースを参照することによって、自プロジェクトで発生している問題に類似した事例を探し出す診断手法だ。類似事例における進行過程や対策などを参考にすれば、自プロジェクトの改善に応用できる。

「測定項目による診断」は、定量的

図表2-5●見える化手法とツール

手法	ツール	手法の概要 詳細は【第3章 見える化】を参照
俯瞰	俯瞰図(システム構成、ステークホルダー、スケジュール、要員遷移)	プロジェクトを高い所から俯瞰し、見えにくくなりがちなプロジェクトの全体像を的確に把握する
自己診察	自己評価シート	プロジェクトの主要メンバーが、「自己評価シート」を用いて、自分でプロジェクトに関する問題の有無を診察する
ヒアリング診断	ヒアリングシート	PMOの経験豊富な専門家チームが、チェックシートを使ったヒアリングと、判断の証拠となるプロジェクト・マネジメント資料のチェックを通じて、問題の有無を診察する。その結果、症状が現れていれば、何が問題かを診断し、適切な対応策を指示する。時間と費用はかかるが、正確な診察・診断と適切な対応策をとることができる
事例ベース診断	失敗事例 データベース	対象とするプロジェクトの状況を失敗事例データベースと比較し、類似の状態や現象を見つけ出す。それを参考にすることで、その進行過程や悪化状況を類推することができる
測定項目による診断	管理帳票、EPMツール(自動データ収集分析ツール)、テスト自動化ツール	プロセスの状況の時間的推移、複数の測定データの相互関係を時系列で見ることにより、問題を抽出する。定性的アプローチで仮定した問題の検証や、場合によっては、今後の到達点の推定などが可能になる。そのために、複数の定量データを測定・収集し、加工・組み合わせて評価する

な測定を行い、測定値やその時間的推移を基にプロジェクトの問題を捉える手法である。その測定項目は「測定項目リスト」として整理した。バグ票や進捗管理表などの管理帳票類からデータを収集するほかに、テスト自動化ツールや自動データ分析ツール（EPMツールなど。【第6章 「見える化」に関する研究の解説】を参照）などがある。

それぞれの「見える化」手法によって、すぐに問題が明らかになる場合には、その対応策の例を提示している。例えば自己評価シートやヒアリングシートでは、質問項目ごとに「もしその質問項目の内容に対応する問題が判明した場合、どのように対応すべきか」を記載している。

留意すべきこととして、各手法に示される対応策はプロジェクト・メンバーに問題個所に対する気付きを与えるための情報である点だ。【第4章 言える化】で述べるように、本来は複数の見える化手法を使って得た結果に対し、プロジェクトの特性に応じた「統合的な判断」を行った上で対策を講じる必要がある。

2.2.3 言える化手法

ITプロジェクトにおける問題は、様々な要因による様々な現象として現れてくる。複合的な問題の場合には、単独の見える化手法だけでは症例を特定できないので、統合的アプローチ手法を用いて症例パターンから原因を特定する手法を説明している（詳細は【第4章 言える化】を参照）。

統合的アプローチ手法は、チェックシートを使って得られた定性的な情報や、データ収集ツールなどを使って収集した定量的な情報、過去の失敗プロジェクト事例などを基に、統合的な判断を行う手法である。例えば、チェックリストでの質問事項を担保するような測定項目が測定できているのか、プロジェクトの症状からどのような項目を測定し、分析することで見える化するか、という関係を「症例分類表群」を使って整理している。

2.2.4 直せる化手法

見える化手法や言える化手法でプロジェクトの問題が見えた後、どのようにしてそれらの問題を是正していくかを検討する必要がある。ただし、プロジェクトの問題解決の方法には、決まったやり方はない。なぜなら、プロジェクトの状況に応じて多様な改善方法があり得るからである。

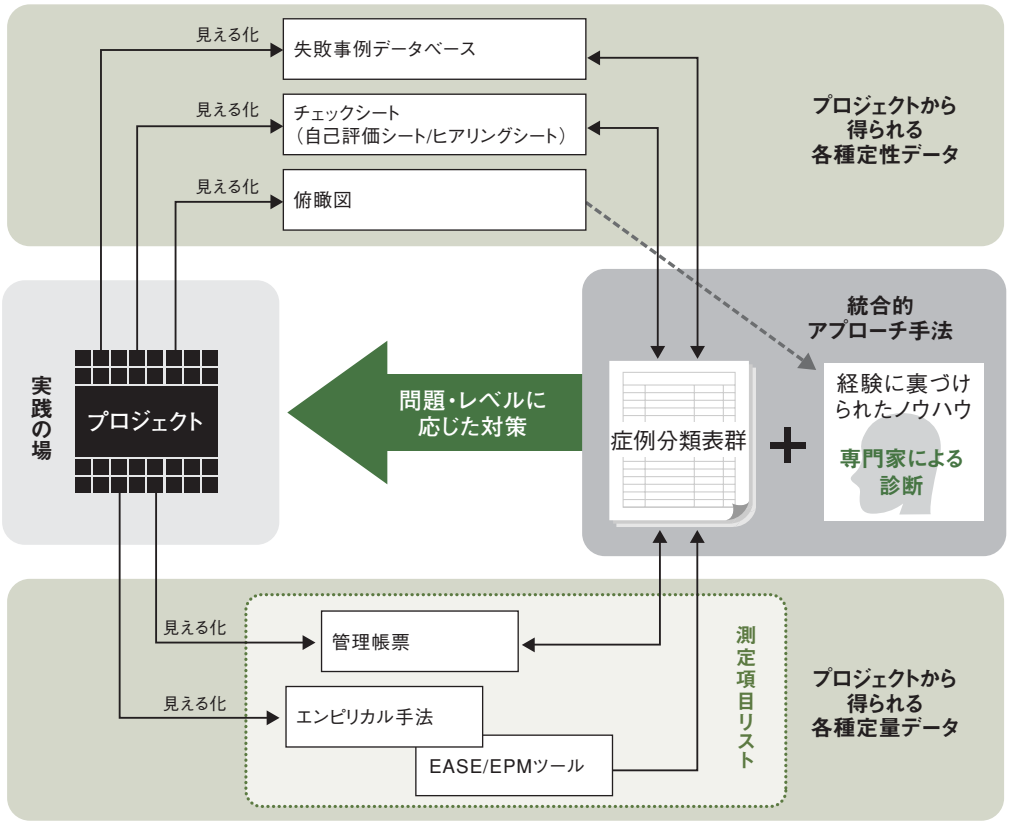
直せる化手法では、問題を改善する活動について、どのような考え方があるかを整理し、改善活動のパターンを定義する。

2.2.5 見える化の手法とツール、適用事例

見える化マネジメントの手法、ツールの全体像を**図表2-6**に示す。実践の場（すなわちプロジェクトの開発現場）に対して、定性的、定量的なアプローチでプロジェクトを見える化する。これらの測定結果を統合的アプローチ（症例分類表群）で分析して問題を特定し、有識者としての経験やノウハウを織り交ぜながら問題やレベルに応じた対策をプロジェクトの開発現場にフィードバックしていく。

定量的アプローチの一つとして、エンピリカル・アプローチ（実証的なアプローチ）と呼ばれるものがある。本書

図表2-6●手法・ツールと見える化イメージ



では、定量データに基づいてソフトウェアの生産性や信頼性向上を行う、エンピリカル・ソフトウェア工学に基づいた測定方法を「エンピリカル手法」と総称している。

これに関連するプロジェクトとして、奈良先端科学技術大学院大学と大阪大学は平成15年度から文部科学省のリーディング・プロジェクト「e-Society基盤ソフトウェアの総合開発」の一環として、EASE（Empirical Approach to Software Engineering）プロジェクトを進めている。EASEプロジェクトでは、「EPMツール」というプロジェクト・モニタリング・ツール群を開発している。EPMツールについては、本書の【6.1 EASEプロジェクトのEPMツールを使ったプロジェクト定量化】、【6.4 コードクローン分析】、【6.5 EASE協調フィルタリングによるプロジェクトの特性予測】を参照してもらいたい。

第3章

見える化

3.1 概要

プロジェクトで発生する問題は、実に様々な現象として現れる。一見して分かりやすい問題としては、「開発したアプリケーション・プログラムが動かない」、「人手不足で十分なテストができない」などがある。

一方、複数の原因が絡んで発生する合併症のような問題、経験がなければ気付かないような問題など、やっかいな場合も少なくない。これらが発生する予兆を早めに掴むことや、発生してしまった問題について的確に現象を把握することは、問題解決のために非常に重要なことである。

しかし、的確な問題の把握は、プロジェクト・マネージャとして経験の浅い者にとっては決して簡単なことではない。

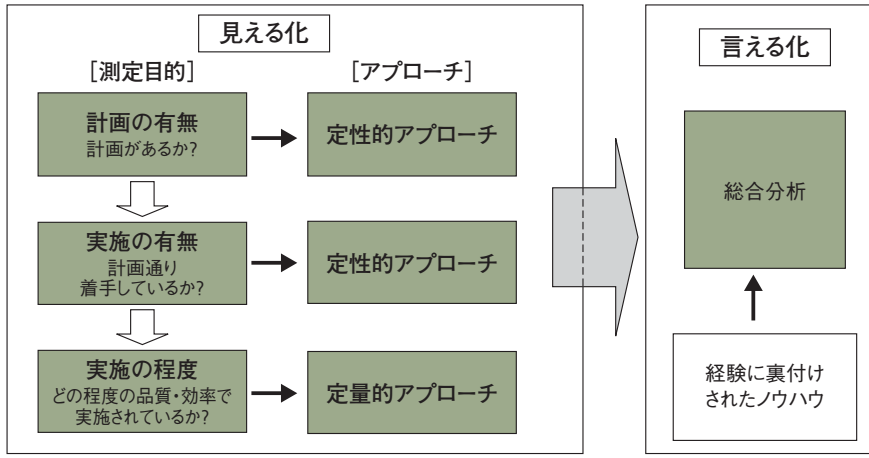
このため、システム開発の現場では、よく「チェックリスト」を使って問題を洗い出したり、「課題管理表」や「バグ管理表」などの管理帳票を使ったりして、定性的に問題の発見・把握に努めてきたことだろう。また、ソフトウェア・エンジニアリングの世界では、テストなどを通して得られたソースコード行数当たりのバグ数などを基に、定量的な品質評価を行ってきた。ITプロジェクトの「見える化」を進めるうえでも、こうした定性的、定量的な手法を組み合わせて使う。

本章では、プロジェクトの状況を捉えるための最初の活動である「定性的・定量的なデータ収集」について説明する。データ収集は、プロジェクトの見える化のために、問題となる現象を察知することに主眼を置いた活動である。

図表3-1のように、プロジェクトの状況を診断するときの基本的なアプローチは3段階ある。まず、ある開発プロセスについて「計画されているかどうか（計画の有無）」というレベル。そもそもプロジェクトで、必要なプロセスが計画されていないことがあるからだ。このようなところからチェックしていく必要がある。

次に「計画通りに着手しているかどうか（実施の有無）」。プロジェクトで必要

図表3-1●見える化のアプローチ



な計画は立てているものの、現場の諸事情により着手できないケースは多い。そこから生じ得る問題を予見することは重要である。

最後に「どの程度の品質・効率で実施されているか（実施の程度）」を見ること。実際、計画通りに作業に着手していても、品質や効率が要求水準に達しているとは限らないからだ。

「計画の有無」や「実施の有無」は、チェックリストなどの定性的アプローチでデータを収集できる。その結果を見ることで、プロジェクト・マネージャには様々な気付きがあるだろう。プロジェクトで起こりつつある問題、あるいは既に表面化した問題に対して、何らかの仮説を立てられることもある。

定性的なアプローチとして、「俯瞰図」、「チェックシート」を使った自己評価、ヒアリング評価の方法について説明する。また、過去の事例を参照することで、プロジェクトの類似性をもって問題を把握する方法について示す。

さらに、ある開発プロセスが「どの程度の品質・効率で実施されているか」を見るために、定量的アプローチで詳細評価を行う。本章では、機械的に測定できる、あるいは人手で収集できる計測項目をまとめた「測定項目リスト」を基に、プロジェクトの定量データの収集手法について説明する。詳細評価では、例えばプロセスの状況の時間的推移や、複数の測定データの相互関係を時系列で見ることが多い。これにより

問題の抽出・特定ができ、定性的アプローチで仮定した問題の検証や、場合によってはプロジェクトの先行きまで推定できる。

定量データの収集には、測定するデータの数・量によっては、人手あるいはツールに相当のコストがかかる。プロジェクト開始時にどこまで管理するかを決め、それに必要なコストを投入する必要がある。

3.2 俯瞰図を用いた見える化

広辞苑によると、俯瞰とは、「高いところから見おろすこと」となっている。だが多くの場合、「俯瞰」という言葉には、ただ高いところから見おろすというだけでなく、地面にいたら把握しにくい事柄、すなわち、それぞれの個別事情に埋没して見えにくくなった全体的な状況を、高いところからより的確に把握するという積極的な意味がある。

プロジェクト・マネジメントにも同じことが言える。現場にいると見えづらくなってしまうことがあるのだ。この点を解消するために必要なツールが「俯瞰図」である。

ただし、同じものを見ていても、見る人の関心がどこにあるかで「見えるもの」が異なる。また、どこに関心を持っているかによって、その「見方」も変わる。

プロジェクトごとに個性があるITプロジェクトでは、何が起こるか予測がつきにくい。このため俯瞰図の作成では、「何がドミナント・アイテムになりそうか」、「リスクを事前に阻止するために、何をどのように見なければならないか」に関心を絞って考察する必要がある。それを繰り返し、自分で納得できるまで俯瞰図を練り上げていく。この段階では、プロジェクト・マネージャはドミナント・アイテムを把握できているはずだ。

ITプロジェクト・マネジメントに特に役立つ俯瞰図は次の4つである。

①システム構成俯瞰図（開発システム構成の全体像やドミナント・アイテムが分かる）
②ステークホルダー俯瞰図（関係するステークホルダーの全体像が分かる。プロジェクト体制図を付け加えると、プロジェクト推進体制の全体像が分かる）
③スケジュール俯瞰図（全体スケジュールの内、重点的にマネジメントすべきクリティカルなスケジュールが分かる）
④要員遷移俯瞰図（フェーズごとにキーパーソンのスケジュール、作業内容が分かる）

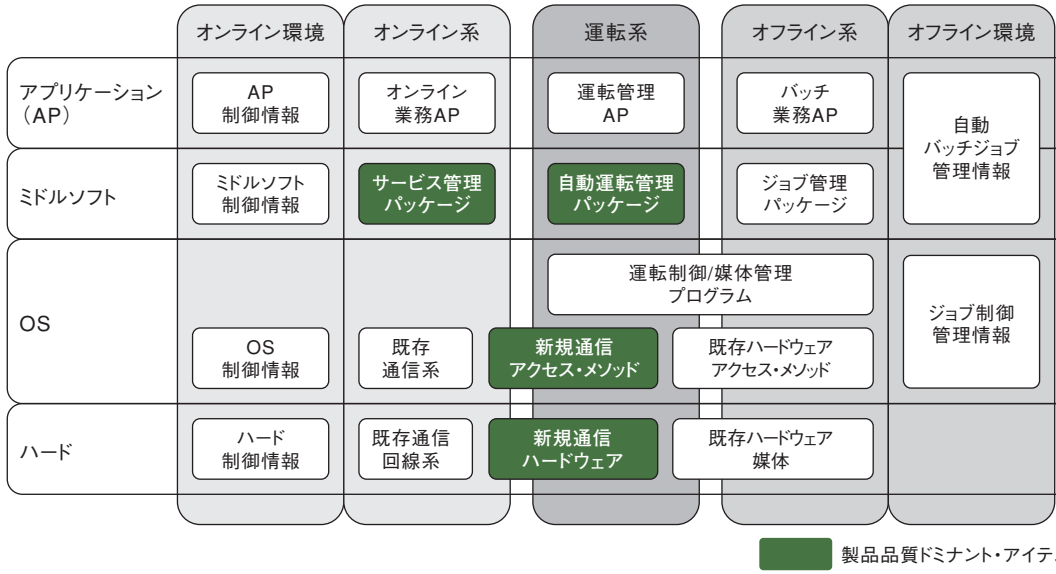
3.2.1 システム構成俯瞰図

今回開発するシステムが、関連するシステムのどこに位置付けられているかを俯瞰するために作成する。この俯瞰図を作成することによって、他のシステムとの関連性が分かるので、プロジェクトとして今後検討していかなければならない項目（例えばシステム間での結合テストにおける優先度やテストの作業項目）を洗い出したり、システムのボトルネックなどを検討したりすることができる。

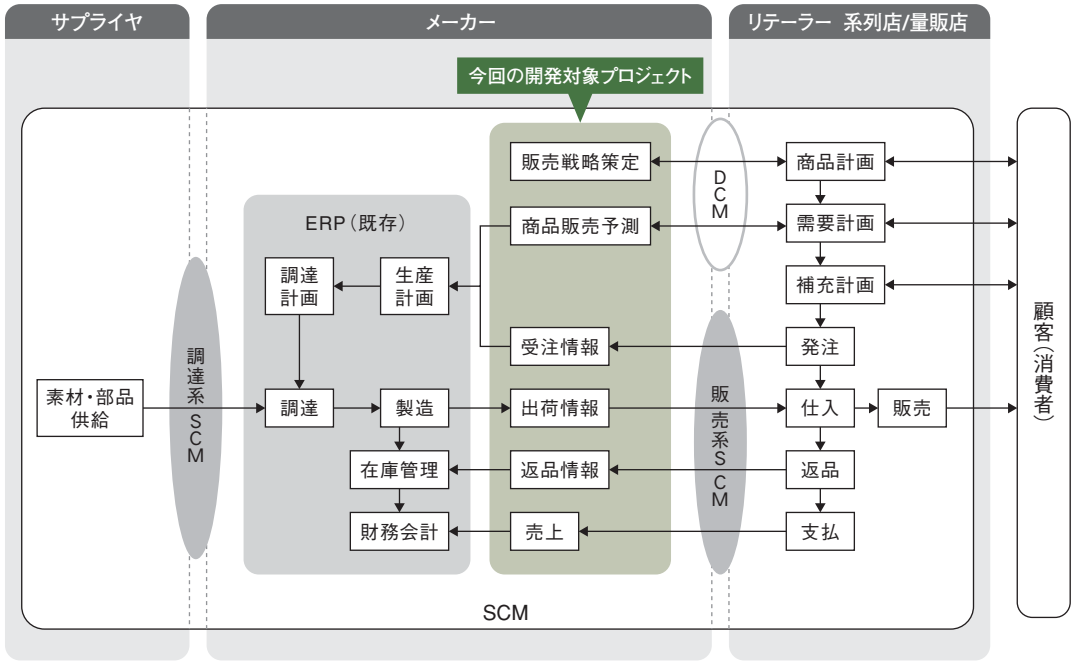
システム構成俯瞰図を作成する際の基礎データとしては、プロジェクト計画書の標準的なデータ項目を用いるとよい。例えば、プロジェクトの特性（アーキテクチャなどのシステム特性、要求仕様明確度などのユーザー要求管理、要員のスキルなど）、システム規模（ライン数やファンクションポイント数など）、工期、工数などが利用できる。これらには、ITプロジェクトのドミナント・アイテムを見る上で重要なデータが含まれており、システム俯瞰図の作成に活用できる。

一般的には、**図表3-2**のようにサブシステム内部のシステム構成俯瞰図を作成すればよい。ただし、企業内あるいはサプライチェーンの中で開発システ

図表3-2●システム構成俯瞰図の例（内部）



図表3-3●システム構成俯瞰図の例（外部）



DCM：デマンドチェーン・マネジメント　　SCM：サプライチェーン・マネジメント

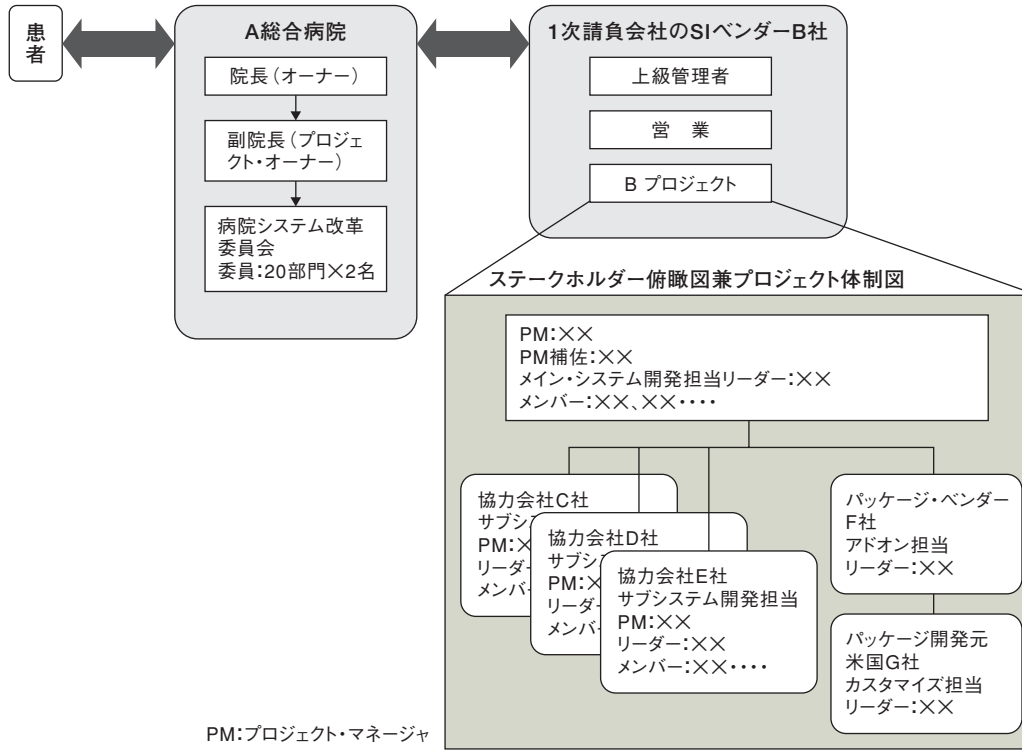
ムの位置付けを把握したい場合は、**図表3-3**のような、開発対象システムの外側にあるシステムを含めた俯瞰図を作る必要もある。

3.2.2 ステークホルダー俯瞰図

プロジェクト・マネジメントに関係するステークホルダーは、最近その数が多くなり、しかも影響力を増している。また、プロジェクトの規模が大きくなると、開発ベンダー側も1次請負会社（コントラクタ）、2次請負会社、さらに開発ベンダー、ツール・ベンダー、ハードウェア・ベンダーなど、複数の企業による階層的な構造になることが多い。関係するステークホルダーが多くなるほど、コミュニケーション・マネジメントに問題が発生しやすくなるばかりでなく、意志決定が遅れたり、責任分担があいまいになったりと、様々な問題が発生し得る。

そこで**図表3-4**に示すように、顧客や1次請負会社、調達先といったステークホルダーの全体像を俯瞰するため

図表3-4●ステークホルダー俯瞰図の例



の「ステークホルダー俯瞰図」を作成することが重要になってくる。

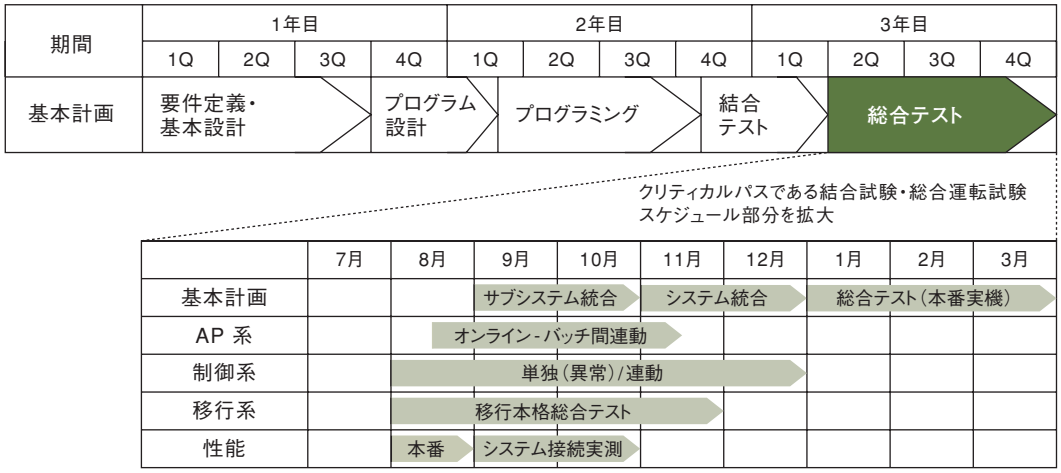
特に、プロジェクトの責任者であり最終意思決定者であるオーナーを明確にすることや、キーパーソンが誰であるのかを確認することは重要である。プロジェクト体制図のポジションごとに、その役割と具体的な担当者を記入するとキーパーソンを把握しやすい。

このステークホルダー俯瞰図を作成する過程で、様々な関係者とコンタクトできる。プロジェクトの外にいるPMOなどの専門家チームは、ぜひ人間関係の構築に留意してステークホルダー俯瞰図を作成してもらいたい。俯瞰図を作成する段階でプロジェクト・メンバーと関係を築くことにより、その後のヒアリング作業や定量データの収集依頼などがしやすくなるはずだ。

3.2.3 スケジュール俯瞰図

複数のサブシステムを含むようなプ

図表3-5●スケジュール俯瞰図の例



ロジェクトの場合、各サブシステムの詳細スケジュールはそれぞれの担当者が作成して管理している。だが、この詳細スケジュールが百本以上あるような非常に大きなプロジェクトでは、プロジェクト・マネージャがすべての詳細スケジュールをマネジメントするのは不可能である。

そこで、すべてのサブシステムの作業スケジュールを全体的に見渡せる「スケジュール俯瞰図」を作成する必要がある。

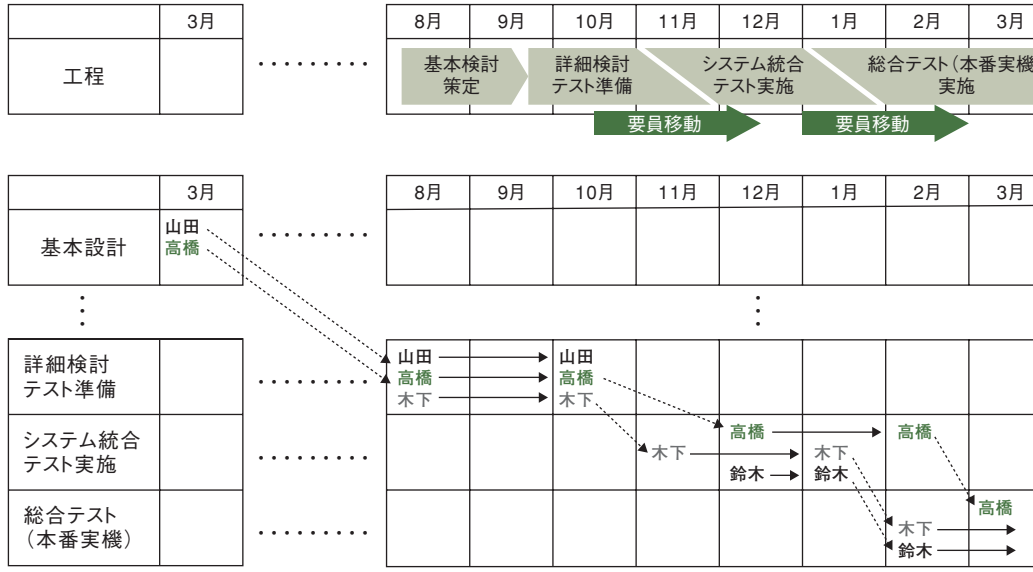
スケジュール俯瞰図は、**図表3-5**のようにプロジェクトの開始から終了までの全体スケジュールを俯瞰するために作成する。詳細スケジュールだけを見ていると、「木を見て、森を見ず」の近視眼的なマネジメントに陥ってしまい、関連サブシステムやサービスインのスケジュールを考慮した全体的なマネジメントができなくなってしまうためである。

この俯瞰図を作成し、プロジェクト・マネージャが最も注意を払うべきクリティカルパスのスケジュールを明確にすることは、作業の優先順位の判断や事前準備の段取りミスの防止などに有効である。

3.2.4 要員遷移俯瞰図

システム設計者のキーパーソンが統合テストを担当すれば、業務内容やシステムの全容を把握しているため、テストの品質を高められるだけでなく、テ

図表3-6●要員遷移俯瞰図の例



スト効率も改善できる。

しかし、プロジェクトの現場では、単体テストが終わらないうちに結合テストに入ることも少なくない。これは、「プロジェクト・メンバーは複数いるので、作業工程を同時並行で進められるはず」という考えに基づいている。

この方法の問題点は、プロジェクトの下流工程で進捗が悪くなり始めると、本来なら並行して進んでいる別の作業を担当するはずだった優秀な人材を、遅れている作業に投入しがちなことである。その結果、キーパーソンとなっているメンバーに高い負荷を強いることになりやすい。

このような問題を避けるために「要員遷移俯瞰図」を作成する。**図表3-6**のように、上流工程でシステム設計を担当したキーパーソンが、中流工程や下流工程で担当すべき作業項目を遷移図の形で整理し、全体を俯瞰する。これにより、要員の適正配置、過不足の点検に利用することができる。

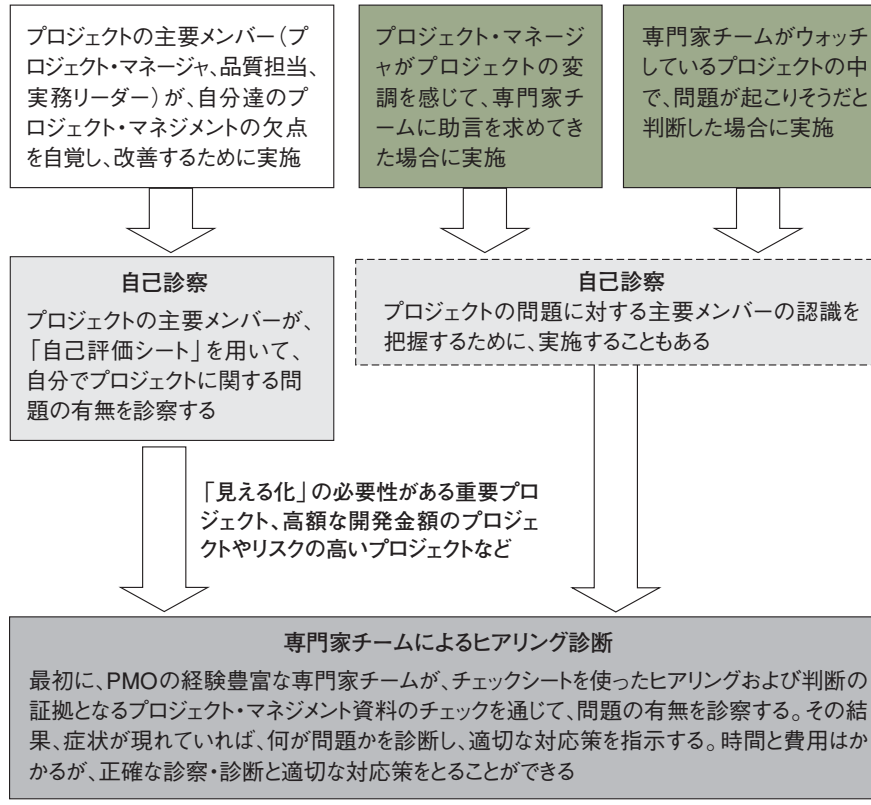
3.3 チェックシートを用いた見える化

経験則から導いた数十個のチェック

項目があれば、プロジェクトに問題が起こっているかどうかを定性的に把握することができる。そのためにSECは、プロジェクト・マネージャなどが自己診察に用いる「自己評価シート」と専門家チーム（例えばPMO）によるヒアリング診断で用いる「ヒアリングシート」の2種類を開発した。

図表3-7にチェックシートを用いた「見える化」の全体の流れを示す。チェックシートを用いた見える化を行うきっかけとしては、①プロジェクトの主要メンバーがプロジェクト・マネジメントの欠点を自覚して改善するケース、②プロジェクト・マネージャがプロジェクトの変調を察知して専門家チームに助言を求めるケース、③専門家チームがウォッチしているプロジェクトで、問題

図表3-7●チェックシートを用いた見える化の流れ



が起こりそうだと判断したケース、がある。

①のケースでは自己診察（「自己評価シート」を利用）、②と③のケースでは専門家チームによるヒアリング（「ヒアリングシート」を利用）を行い、問題の見える化を行う。

3.3.1 「自己評価シート」の構成

自己診察を行う場合に利用する「自己評価シート」のイメージを**図表3-8**に示す。自己評価シートには、全部で40の質問項目（チェック項目）があり、「S」で始まるチェック項目番号（No.）が振られている。それらのチェック項目に順次回答していくことにより、知識エリアごとの診察結果が自動的にグラフ化される仕組みとなっている。

回答は、プロジェクト・マネージャ、品質担当、実務リーダーが協力して作成してもらいたい。3者がそれぞれ回答すべき項目を示しているので、該当するところに回答すればよい。

自己評価シートの全体は、【付録2 チェックシート】に掲載する。またこのチェックシートの電子ファイル（Excelファイル）は、SECのホームページからダウンロードできる。

自己評価シートの使い方は後述するとして、自己評価シートに含まれる項目の概要を説明したものが**図表3-9**である。ただし、「平均頻度」から「最大評点」までの項目は表計算上の内部集計に関するものなので、SECのホームページからダウンロードしたチェックシートを使う場合、回答者は意識する必要はない。

図表3-8において、「評価レベル」の欄には「プロジェクト・マネージャ」、「品質担当」、「実務リーダー」という役割ごとにスコアを入れる部分がある。この値の付け方で留意すべき点は、回答者がたとえ「大丈夫」と回答しても、どのように大丈夫なのかを確認しても

図表3-8●自己評価シートのイメージ

自己評価シート											
					評価レベル						
No.	知識エリア	チェック項目	評価基準	マネジメントに当たってのヒント	プロジェクト・マネージャ		品質担当		実務リーダー	
S1	統合	プロジェクト計画が作成され、レビューされているか？	プロジェクト計画書は作業着手から規定期間以内に作成され、レビューされていること	期間が設定されていない場合は、作成期間を設定すること	○	1	○	2		
S2	統合	構成管理の対象として、納入物件及び必要な作業成果物が明確になっているか？	顧客との間で成果物のイメージが明確になっていること	顧客との間で成果物のイメージが明確でないとその後の作業の割り直しや修正の見積もりなどで大きな問題に発展する可能性がある	○	2			○	3
S3	統合	すべての作業項目に対して、スケジュールを立てているか？	スケジュールに記載のない作業項目がないこと	クリティカルパスを明確にすることによって、遅れてはならない作業の重点管理を行えるようになる	○	1				

図表 3-9●自己評価シートの項目説明

項目	項目説明	
No	チェック項目の番号。Sで始まる連番	
知識エリア	チェック項目が属する知識エリア名。知識エリア名は次の15種類。PMBOKにある9つの知識エリア（統合、スコープ、タイム、コスト、品質、人的資源、コミュニケーション、リスク、調達）に加え、PMBOKに不足しているヒューマンウェア部分として「顧客」、「技術」、「組織」、「基本動作」、「モチベーション」の5つ、さらに、プロジェクトの下流工程における監視コントロール・プロセスに注目した「課題管理」	
チェック項目	プロジェクトの状況をたずねる質問項目。合計の質問数は40	
評価基準	評価レベルを判断するための具体的な基準やチェックすべき具体例	
マネジメントにおけるヒント	自己評価する際のポイント、チェック項目の意味や強化すべきマネジメントを気付かせるヒント	
対策案	そのチェック項目の評価が低い場合に取るべき対策の例	
評価レベル	プロジェクト・マネージャ、品質担当、実務リーダーが、各チェック項目に対する評価を3段階で記入する。評価基準は図表3-10を参照	
平均頻度	プロジェクト・マネージャ、品質担当、実務リーダーそれぞれの評価レベルを平準化して点数化したもの。点数が大きいほど、その項目で問題が発生していることを表している。評価レベル3なら0、評価レベル2なら0.5、評価レベル1なら1に置き換えて平均を求めた値が平均頻度となる	
影響度	チェック項目自体が持つ問題の影響度を表す。チェック項目ごとに大（1）、中（0.8）、小（0.6）の3段階で定められている。影響度の大中小は以下の基準で付けている	
	大	プロジェクト全体計画に影響する項目か、当然行われるべき基本事項
	中	下流工程で特に重要視
	小	それ以外
評点	平均頻度 × 影響度の数値	
評点レベル	平均頻度 × 影響度の数値評点を5、4、3、2、1、0 の6段階にレベル分けしたもの。 評点×4倍の小数点切り上げの数値を評点レベルとする	
最大評点レベル	影響度によって評点レベルの上限を設けている	
	影響度1	最大評点レベル5
	影響度0.8	最大評点レベル4
	影響度0.6	最大評点レベル3
判定	評点レベルから判定を算出する	
	評点レベル 5	A
	評点レベル 4	B
	評点レベル 3	C
	評点レベル 2	D
	評点レベル 1	E
	評点レベル 0	空白

平均頻度以降の項目は、付録2には含まれていない。SECのWebサイトからダウンロードできる電子ファイル（Excelファイル）には含まれている

らう必要があることだ。

3.3.2 自己診察の実施方法

(1) チェック項目への回答方法

プロジェクト・マネージャは、自己評価シートの横軸にある「評価レベル」内の「プロジェクト・マネージャ」欄に○が付いているチェック項目について、評価レベル1～3の数字を入力する。評価レベルの意味と評価基準は**図表3-10**に示す。

同様に、品質担当は、「品質担当」の欄に○が付いているチェック項目、実務リーダーは、「実務リーダー」の欄に○が付いているチェック項目について、評価レベル1～3の数字を入力する。

(2) 自己診察結果のレーダーチャート

すべてのチェック項目に対する回答が終わると、**図表3-11**のように自己診察の結果集計表と15の知識エリアの評点レベルで構成されるレーダーチャートがサマリーシート（同じExcelファイル内の別のワークシート）に自動的に表

図表3-10●自己診察の評価レベル

評価レベル	評価基準
1	チェック項目のマネジメント方法を知らない
2	チェック項目のマネジメントとして何をしなければならないかは分かっているが、種々の事情によってうまくできていない
3	チェック項目のマネジメントとして何をしなければならないかが分かっており、うまくできている

自己評価シート使用時の留意事項

自己評価シートを使用する際は、次のことに留意する。

① 主要メンバーの主観に基づく評価なので、事実は異なる可能性がある。
② 限られた時間で実施するので、シートの質問項目の網羅性に限界がある
③ 回答者自身がチェックシートの診察結果だけを見て治療することは、逆に新たな問題を抱え込む恐れがある
④ プロジェクトを取り巻く環境が激しく変化しているために、固定化したチェック項目に頼り過ぎると、間違った安心感を生むリスクがある

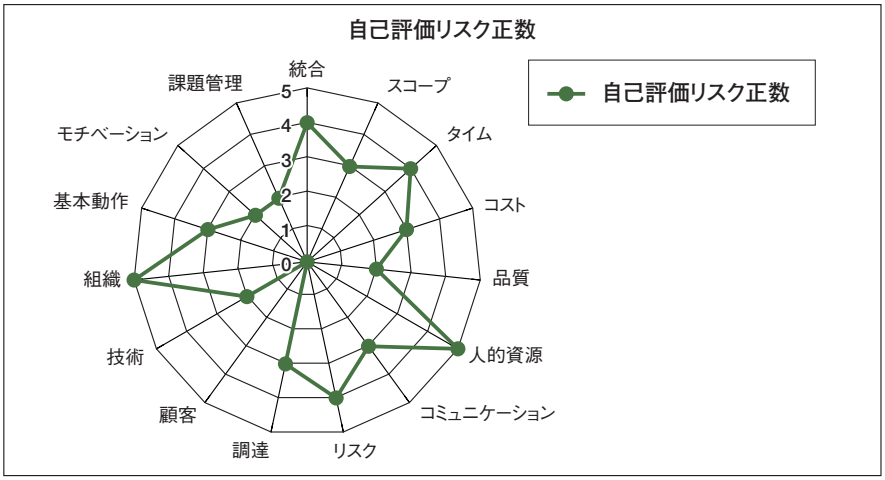
自己評価シートのアップデート方法

自己評価シートは、アップデート作業を行わないと有効性を失い、すぐに使われなくなる。アップデートは次のように実施する。

① 自組織に定着したチェック項目を削除する
② 自組織に新たに強化したいチェック項目を追加する
③ プロジェクトを取り巻く環境変化に対応するため、自組織で気付いた新たなチェック項目を追加する
④ プロジェクトを取り巻く環境変化に対応するため、SECから提供される新たなチェック項目について、自組織にとって要否を判断し、必要であれば追加する

図表3-11●自己診察結果の表とレーダーチャート

項番	知識エリア	自己評価				
		評点レベル合計	最大評点レベル	比率	リスク数	リスク正数
1	統合	8	34	0.235294	1	4
2	スコープ	4	13	0.307692	2	3
3	タイム	3	20	0.15	1	4
4	コスト	2	6	0.333333	2	3
5	品質	3	4	0.75	3	2
6	人的資源	0	12	0	0	5
7	コミュニケーション	4	12	0.333333	2	3
8	リスク	3	15	0.2	1	4
9	調達	2	4	0.5	2	3
10	顧客	3	3	1	5	0
11	技術	2	3	0.666667	3	2
12	組織	0	5	0	0	5
13	基本動作	5	10	0.5	2	3
14	モチベーション	8	15	0.533333	3	2
15	課題管理	3	5	0.6	3	2



示される。レーダーチャートを利用することで視覚的かつ直感的に問題となる知識エリアを把握することができる。自己診察結果の表の項目説明は**図表3-12**の通りである。リスク正数の値がレーダーチャートに表示される。

このレーダーチャートの例では、「人的資源」や「組織」に問題が潜んでいる可能性は低いが、「顧客」には問題が潜んでいる可能性が特に高いことが分か

図表3-12●自己診察結果の表の項目説明

項目	説明
知識エリア	サマリー対象の知識エリアを示す
評点レベル合計	自己評価シートの評点レベルを知識エリアごとに集計したもの
最大評点レベル	自己評価シートの最大評点レベルを知識エリアごとに集計したもの
比率	評点レベル合計/最大評点レベル
リスク数	比率をレベル分けしたもの ・比率1ならリスク数は5 ・比率1以外なら、リスク数は比率×4 の小数点切り上げの値
リスク正数	リスク数の5の補数。レーダーチャートの表示用の数字

る。このような場合には、評点レベルが低い知識エリアのチェック項目を点検し、必要に応じて次ステップの「専門家チームによるヒアリング診断」の実施を検討する必要がある。

(3) 自己診察結果の活かし方

自己診察後、不得手な領域が明確になるので、自己評価シートの「対応案」を参考に改善するよう努力してもらいたい。

自己診察を毎月実施すると、改善状況を時系列的に評価できる。引き続き専門家チームによるヒアリングを行う場合には、自己評価シートの写しを専門家チームに渡す。

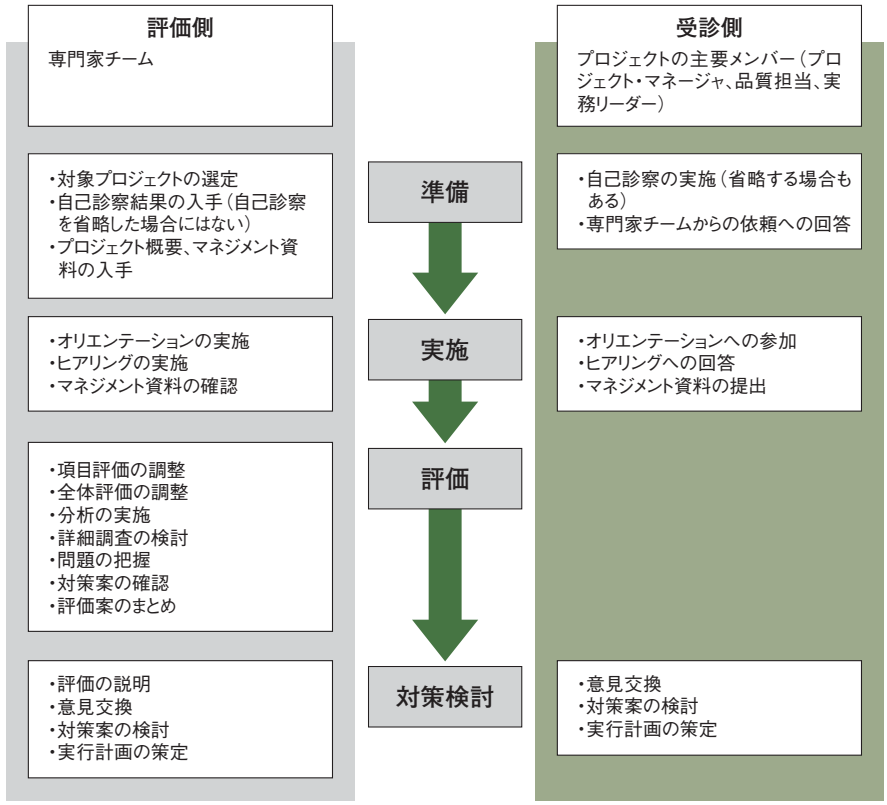
3.3.3 専門家ヒアリングの流れ

専門家チームによるヒアリング診断の流れは**図表3-13**の通りである。評価側（専門家チーム）と受診側（プロジェクト・マネージャら）が協力して効率的に診断を行う。

まず準備段階では、自己診察（実施している場合）の結果の入手やプロジェクト・マネジメント資料の入手を双方が協力して行う。実施段階では、評価側が「ヒアリングシート」のチェック項目に基づき順次評価を行う。ヒアリング中には質疑応答だけでなく、エビデンス（プロジェクト資料）を確認しながら、事実を正確に把握することに努める。

評価段階では、評価側がヒアリング結果をチーム内で討議し、評価案としてまとめる。最後に対策検討段階では、双方が評価案に対して共通認識が得られるまで議論し、その結果を踏まえて

図表3-13●専門家ヒアリングの流れ



対策案の立案と実行計画の策定を行う。

3.3.4 ヒアリングシートの構成

専門家チームによるヒアリング診断を行う場合に利用する「ヒアリングシート」のイメージを**図表3-14**に示す。

ヒアリングシートには全部で85の質問項目（チェック項目）があり、「H」で始まるチェック項目番号（No.）が振られている。その項目に沿って順次チェックすることにより、知識エリアごとの評価結果が自動的にグラフ化される仕組みになっている。ヒアリング対象者は、プロジェクト・マネージャ、品質担当、実務リーダーであり、各々該当する項目（ヒアリングシートの「評価

図表3-14●ヒアリングシートのイメージ

レベル」内で○印が付いている項目）をヒアリングする。

ヒアリングシートの全体は【付録2 チェックシート】に掲載する。ヒアリングシートの電子ファイル（Excelファイル）は、SECのホームページからダウンロードできる。

ヒアリングシートの使い方は後述するとして、ヒアリングシートに含まれる項目の概要を説明したものが**図表3-15**である。「平均頻度」から「最大評点」までの項目は表計算上の内部集計に関するものなので、SECのホームページからダウンロードしたチェックシートを使う場合、回答者は意識する必要はない。

「評価レベル」欄には「プロジェクト・マネージャ」「品質担当」「実務リーダー」という役割ごとに評価レベルを記入する部分がある。この値の付け方で留意すべき点は、回答者が「大丈夫」と回答しても、どのように大丈夫なのかをヒアリングで確認してもらう必要があることだ。

3.3.5 ヒアリング診断の実施方法

(1) 評価する専門家チームの構成

ヒアリング診断を行う専門家チームのメンバーは、**図表3-16**に示すように2〜3人の構成とする。専門家チームの中心となるのはヒアリング実施者であり、チェック項目の質問を行うとともに、必要に応じてエビデンス（プロジェクト資料）の確認もするため、他の作業（記録者）との兼務は無理である。ヒアリング実施者には豊富なプロジェクト・マネジメント経験が要求される。

(2) 実施前の対象プロジェクト・マネージャへの依頼

図表3-15●ヒアリングシートの項目説明

項目	項目説明	
No.	チェック項目番号。H で始まる連番	
知識エリア	チェック項目が属する知識エリア名。知識エリア名は次の15種類。PMBOKにある9つの知識エリア（統合、スコープ、タイム、コスト、品質、人的資源、コミュニケーション、リスク、調達）に加え、PMBOKに不足しているヒューマンウェア部分として「顧客」、「技術」、「組織」、「基本動作」、「モチベーション」の5つ、さらに、プロジェクトの下流工程における監視コントロールプロセスに注目した「課題管理」	
チェック項目	プロジェクトの状況をたずねる質問項目。合計の質問数は85	
評価基準	評価レベルを判断するための具体的な基準やチェックすべき具体例	
個別のヒアリング要領	そのチェック項目をヒアリングする際のポイント、チェック項目の意味やプロジェクト・マネジメント上の心得	
エビデンス・確認方法	チェック項目を判断する際に証拠となる資料とその確認方法	
対策案	そのチェック項目の評価が低い場合に取るべき対策の例	
評価レベル	専門化チームが、各チェック項目に対する評価を5段階で記入する。評価基準は図表3-20を参照	
平均頻度	プロジェクト・マネージャ、品質担当、実務リーダーそれぞれの評価レベルを平準化して点数化したもの。点数が大きいほど、その項目で問題が発生していることを表している。評価レベル5なら0、評価レベル4なら0.3、評価レベル3なら0.5、評価レベル2なら0.8、評価レベル1なら1に置き換えて平均を求めた値が平均頻度となる	
影響度	チェック項目自体が持つ問題の影響度を表す。チェック項目ごとに大（1）、中（0.8）、小（0.6）の3 段階で定められている。影響度の大中小は以下の基準で付けている	
	大	プロジェクト全体計画に影響する項目か、当然行われるべき基本事項
	中	下流工程で特に重要視
	小	それ以外
評点	平均頻度 × 影響度の数値	
評点レベル	評点を5、4、3、2、1、0 の6 段階にレベル分けしたもの。 評点×4 倍の小数点切り上げの数値を評点レベルとする	
最大評点レベル	影響度によって評点レベルの上限を設けている	
	影響度1	最大評点レベル5
	影響度0.8	最大評点レベル4
	影響度0.6	最大評点レベル3
判定	評点レベルから判定を算出する	
	評点レベル 5	A
	評点レベル 4	B
	評点レベル 3	C
	評点レベル 2	D
	評点レベル 1	E
	評点レベル 0	空白

平均頻度以降の項目は、付録2には含まれていない。SECのWebサイトからダウンロードできる電子ファイル（Excelファイル）には含まれている

図表3-16●専門家チームの構成

役割	兼務の可能性	経験または能力
ヒアリング実施者	兼務は原則として無理	豊富なプロジェクト・マネジメント経験を持っていること。プロジェクトで「今起こっている問題」および「今後起こる問題」を見通す高度な能力が求められる
ヒアリング対象者の表情・仕草などの観察者	ヒアリング実施者の兼務も可能	
ヒアリング内容の記録者	兼務は無理	―

図表3-17●プロジェクト概要記述用紙の項目説明

項目	説明
開発システム	開発対象のシステムの5W1H（目的、ユーザー、利用場面・場所、入出力、システム形態/処理内容）を簡潔に記述する
特徴	開発対象システムにおいて特徴的な事柄を記述する。具体的には、新技術・未経験技術の適用有無、新規顧客か否か、新規業務/業務改革の有無、新しい開発体制（新しいプロジェクトマネージャ、新しい協力会社）、品質・コスト・納期に対する制約の度合、大規模処理の有無、パッケージ利用の有無、特徴的な開発ツールの利用有無、遠隔地開発の有無、現行システム保証の度合、開発において制約となる事項（開発手法、モジュール共通化の要請、既存システムの再利用など）の有無などがある
プラットフォーム	OS、DBMS など開発対象システムのプラットフォームを記述する
プログラム開発規模	開発対象システムのステップ規模、FP など規模を表す数字を記述する
開発時期	開発開始時期と開発終了予定時期を記述する
開発期間	開発期間を記述する
開発形態	最終顧客と開発に関わる組織を記述する。請負開発が多重構造になっている場合は、下請け構造を全て記述する
プロジェクト要員数（延べ）	プロジェクトに関わる延べ要員数を記述する
プロジェクト要員数（現時点）	現時点でプロジェクトに関わる要員数を記述する
現在のフェーズ	現在の進捗状況が、スケジュール上のどこに当たるのかを記述する
あなたが感じている課題	現在の課題認識を記述する

①プロジェクト概要記述用紙への記入

ヒアリング対象プロジェクトの概要を把握するため、「プロジェクト概要記述用紙」に開発中のシステムの説明、稼働環境、開発期間、要員数などをプロジェクト・マネージャに記入してもらう。プロジェクト概要記述用紙は、SECのホームページから電子ファイルをダウンロードできる。

プロジェクト概要記述用紙の概要を

図表3-18●プロジェクト概要記述用紙への記述例

項目	内容
開発システム	インターネット/コンテンツ・サービス・プロバイダが提供するサービスの使用料に応じて利用者に課金し、料金を代行徴収するシステムを構成するサブシステム（メディエーション・システム）
特徴	オープン技術によるミッションクリティカル・システム。米国ベンチャー企業のパッケージを導入
プラットフォーム	UNIX/Linuxサーバー（DBはMySQL）
プログラム開発規模	約5万ステップ相当（パッケージ、オープンソース・ソフト使用のためステップ換算は困難）
開発時期	2001年～2002年
開発期間	6カ月
開発形態	コンピュータ・メーカーによる請負開発。多重下請け構造
プロジェクト要員数（延べ）	約400人
プロジェクト要員数（現時点）	70人
現在のフェーズ	統合テストの入口
あなたが感じている課題	統合テストにいつ入ればいいのか、迷っている

図表3-17に示す。これだけではプロジェクト概要記述用紙の記入イメージをつかみづらいと思われるので、実際の記入例を**図表3-18**に示す。この事例はインターネット・サービスの利用によって生じた料金の代行徴収システムである。プラットフォームはオープン技術を活用したサードパーティのパッケージ・システムをベースにしている。

②ヒアリング時に持参すべきマネジメント資料の一覧リスト

下流工程でヒアリングを実施する場合の良い点は、尋ねたことに正しく答えているかどうかを確認するエビデンス（証拠）が存在することだ。下流工程なら、プロジェクトの成果物として必要な各種の資料が存在するはず。これを確認することでプロジェクトが正しく運営されているかどうかが分かる。もちろん、内容をつぶさに確認する必要はなく、まず存在するかどうかを確認し、必要に応じて内容をチェックすればよい。

エビデンスとなるプロジェクト・マネジメント資料の一覧リストを**図表3-19**に示す。約60の資料をリストアップしている。

③ヒアリングの場所の確保

ヒアリングを実施する場所は、必要なマネジメント資料をすぐに参照できるように、それらが保管されている場所

図表3-19●ヒアリング時に持参すべきマネジメント資料一覧

領域	資料名	領域	資料名
俯瞰図	システム構成俯瞰図	設計関連	顧客への仮リリース仕様書
	ステークホルダー俯瞰図		設計レビュー結果表
	スケジュール俯瞰図		用語集
	要員遷移俯瞰図	コーディング・製造関連	ソースコード
プロジェクト・マネジメントルール関連	プロジェクト計画書		コーディング規約
	リスク管理表		コードレビュー結果表
	責任分担表	テスト関連	テスト計画書
	体制図		プログラム変更表
	WBS		テストケース消化表
	調達計画書		テスト実施手順書
	構成管理計画書		テストケース表
	要員育成目標		テストデータ
	プロジェクト計画書レビュー結果表		テストツール
			テストレビュー記録
	変更管理ルール		テスト結果報告書
	構成管理ツール		障害表、バグ表
	構成管理ルール・手順書		リリース管理フロー
各種の管理帳票・報告書	課題管理表		リリース手順書
	リスク管理表	スケジュール関連	マスタースケジュール
	変更管理表		チーム別スケジュール
	進捗報告書		テスト・移行スケジュール
	顧客への報告書		要員山積み表
	プロジェクト・メンバーの勤務管理表	移行・運用関連	移行計画書
			保守・運用計画書
設計関連	顧客からのRFP		運用設計書（通常、障害）
	要求管理表		移行計画レビュー結果表
	要件定義書		
	基本設計書		運用計画レビュー結果表
	現行機能仕様書	契約書類	各種契約書
	重要機能リスト		見積もり書
	（上流工程担当者からの）引き継ぎ資料		協力会社選定理由書
	ドキュメントの雛形		協力会社への発注書、作業依頼内示書
	顧客との要求仕様レビュー記録表	議事録	各種の議事録

(つまりプロジェクトの作業場所）に近いところが望ましい。

(3) 自己診察結果の把握

自己診察はプロジェクト・マネージャ、品質担当、実務リーダーたちが、自分達のプロジェクト・マネジメントの欠点を自覚するために実施するもので、すべてのプロジェクトで実施するわけではない。しかし、専門家チームによるヒアリング診断の前に、現場の判断で実施している場合がある。

事前に自己診察を行っている場合、専門家チームはその診察結果を取り寄せる。プロジェクトの主要メンバーが、プロジェクトの状況をどのように捉えているかを事前に把握しておく。

(4) ヒアリング前のオリエンテーション

ヒアリングを実施するに当たって、10分程度でヒアリング対象者にオリエンテーションを実施する。これから2時間ぐらいかけてヒアリングすることの概要や、ヒアリングの目的を説明する。また、「プロジェクトで当然できているべき基本的なことも改めてヒアリングするため、当たり前だと思っても気分を害さずに答えていただきたい」など、ヒアリングをスムーズに進めるために必要と思われることを説明しておく。

オリエンテーションで実施する主な内容は以下の6つである。

①ヒアリング目的の確認
②専門家チームの自己紹介
③ヒアリング対象者の自己紹介
④ヒアリングの進め方の説明
⑤ヒアリング時間（約2時間）の使い方
⑥記載されているプロジェクト概要記述用紙を用いたプロジェクト概要の確認

(5) ヒアリングの実施

ヒアリング実施者は、チェックシートのチェック項目について順次ヒアリングを行い、判断の証拠（エビデンス）となるプロジェクト・マネジメント資料を確認する。

ヒアリング内容の記録者は、ヒアリングの回答内容とエビデンスの確認結果を記録していく。ヒアリング対象者の表情・仕草などの観察者は、回答者が話を聞いたり話したりするときの表情や仕草を記録していく。

ヒアリング実施者はヒアリング項目を読み上げた後、その質問の意図を理解してもらうために具体例などを織り込んで分かりやすく説明する。また、疑問に思うことや違和感があるときは、納得がいくまで質問を繰り返し、深堀りしていくことが必要である。そのようなとき、何かしら問題が潜んでいることが多い。

(6) 各チェック項目のレベル評価

ヒアリングの実施後、各チェック項目に対して、専門家チーム内の意見を調整してレベルを評価する。

各チェック項目は5段階で評価する。レベル1が最も悪く、レベル5が最も良い。さらに、なぜヒアリング時にそのように判断したかもメモ欄に記入するなどして、後でヒアリングの状況が思い出せるようにする。各レベルの評価基準を**図表3-20**に示す。

(7) ヒアリング結果のレーダーチャート

すべてのチェック項目に対する回答が終わると、ヒアリング結果集計表と15の知識エリアの評点レベルで構成されるレーダーチャートがサマリーシート（同じExcelファイル内の別のワークシート）に自動的に表示される（**図表3-21**のグレーの線）。レーダーチャートを利用することで視覚的かつ直感的に問題となる知識エリアを把握することができる。ヒアリング結果の表の項目説明は**図表3-22**の通りである。リスク正数の値がレーダーチャートに表示される。

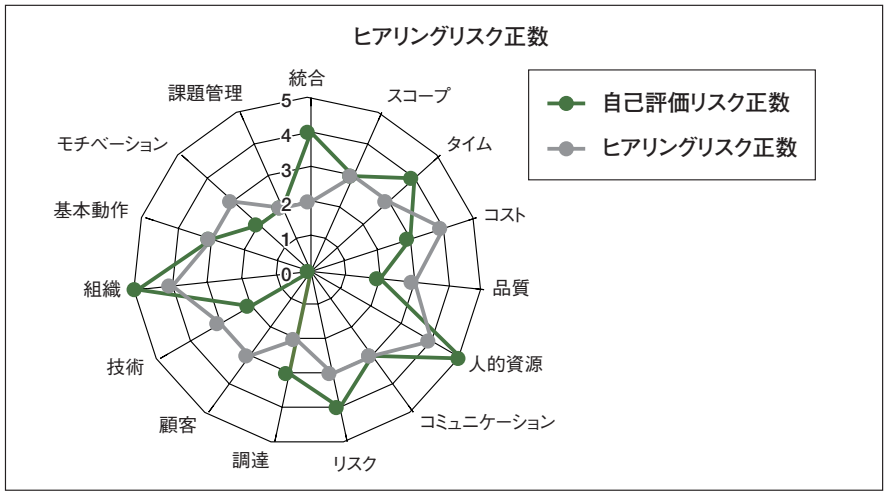
図表3-21のように、自己診察した場合には、自己診察とヒアリングの結果の表と自己診察とヒアリングの結果を重ねたレーダーチャートを表示する。自己診断の結果とヒアリングの結果をレーダーチャートで比較することにより、プロジェクト・マネージャによる自己診断とヒアリング実施者による診断とに差異があるかどうかを見極められる。プ

図表3-20●ヒアリングでの評価レベル

評価レベル	評価基準	診断
1	質問領域のプロジェクト・マネジメントが分かっていない。しかも、的外れなマネジメントを実施	プロジェクト・マネジメントの本質領域である場合は、非常に大きな問題である。 プロジェクト・マネージャの早期交代が必要
2	質問領域のプロジェクト・マネジメントが分かっていないまま、勘と度胸のマネジメントを実施	症例に応じた対応策が必要
3	質問領域のプロジェクト・マネジメントは分かっているが、ほとんどできていない	本来のプロジェクト・マネジメントに戻すために、マネジメント方法を指導する。 さらに、想定していない問題が発生したとき、パニックにならないように、症例に応じた対応策を用意させる
4	質問領域のプロジェクト・マネジメントは分かっているが、内部・外部の要因によって、そのすべてはできていない	本来のプロジェクト・マネジメントに戻すために、マネジメント方法を指導する。 内部・外部要因が影響している場合には、その要因を探すとともに、症例に応じた対応策を用意させる
5	質問領域のプロジェクト・マネジメントがほぼ完璧にできている	—

図表3-21●自己診察とヒアリング結果のレーダーチャート

項番	知識エリア	自己評価					ヒアリング				
		評点レベル合計	評点レベル最大	比率	リスク数	リスク正数	評点レベル合計	評点レベル最大	比率	リスク数	リスク正数
1	統合	8	34	0.24	1	4	18	35	0.51	3	2
2	スコープ	4	13	0.31	2	3	7	17	0.41	2	3
3	タイム	3	20	0.15	1	4	11	34	0.32	2	3
4	コスト	2	6	0.33	2	3	1	8	0.13	1	4
5	品質	3	4	0.75	3	2	23	56	0.41	2	3
6	人的資源	0	12	0.00	0	5	2	10	0.20	1	4
7	コミュニケーション	4	12	0.33	2	3	21	48	0.44	2	3
8	リスク	3	15	0.20	1	4	4	10	0.40	2	3
9	調達	2	4	0.50	2	3	7	13	0.54	3	2
10	顧客	3	3	1.00	5	0	10	22	0.45	2	3
11	技術	2	3	0.67	3	2	7	17	0.41	2	3
12	組織	0	5	0.00	0	5	1	12	0.08	1	4
13	基本動作	5	10	0.50	2	3	6	12	0.50	2	3
14	モチベーション	8	15	0.53	3	2	9	18	0.50	2	3
15	課題管理	3	5	0.60	3	2	7	13	0.54	3	2



図表3-22●ヒアリング結果の表の項目説明

項目	説明
知識エリア	サマリー対象の知識エリアを示す
評点レベル合計	ヒアリングシートの評点レベルを知識エリアごとに集計したもの
最大評点レベル	ヒアリングシートの最大評点レベルを知識エリアごとに集計したもの
比率	評点レベル合計/最大評点レベル
リスク数	比率をレベル分けしたもの ・比率1ならリスク数は5 ・比率1以外なら、リスク数は比率×4の小数点切り上げの値
リスク正数	リスク数の5の補数。レーダーチャートの表示用の数字

ロジェクト・マネージャが「問題なし」と判断していた部分が、外部から見ると実は「問題あり」といった見方ができる。

(8) ヒアリングに基づく診断

レーダーチャートを見ることによってプロジェクトにどのような問題があるか、その傾向を知ることができる。また、レーダーチャートを見るまでもなく、ヒアリングを実施する過程でどのような問題があるかを、うかがい知ることは可能である。

しかし、プロジェクトの問題はヒアリングをベースとしたチェックだけでは、まだ主観的な判断になる可能性がある。このため、定量データの測定や過去事例との対比など、統合的な分析を行った上で問題を把握する必要がある（詳細は【第4章 言える化】を参照）。

(9) 対応策の策定

問題が見つかった場合、また、今後問題になりそうなことが見つかった場合は、事前に対応策を検討する。ここではヒアリングシートの「対応案」も参考にする。

対応策を検討する際は、プロジェクトの状況に関するより詳細な情報が必要になることが多い。このため、ヒアリング対象者にも参加してもらうことが望ましい場合もある。

(10) 診断結果と対応策の説明

対応策を策定したら、それをヒアリング対象者全員に説明する。ただし、プロジェクト・マネージャだけに伝える方がよい内容の場合もあるので、どのよ

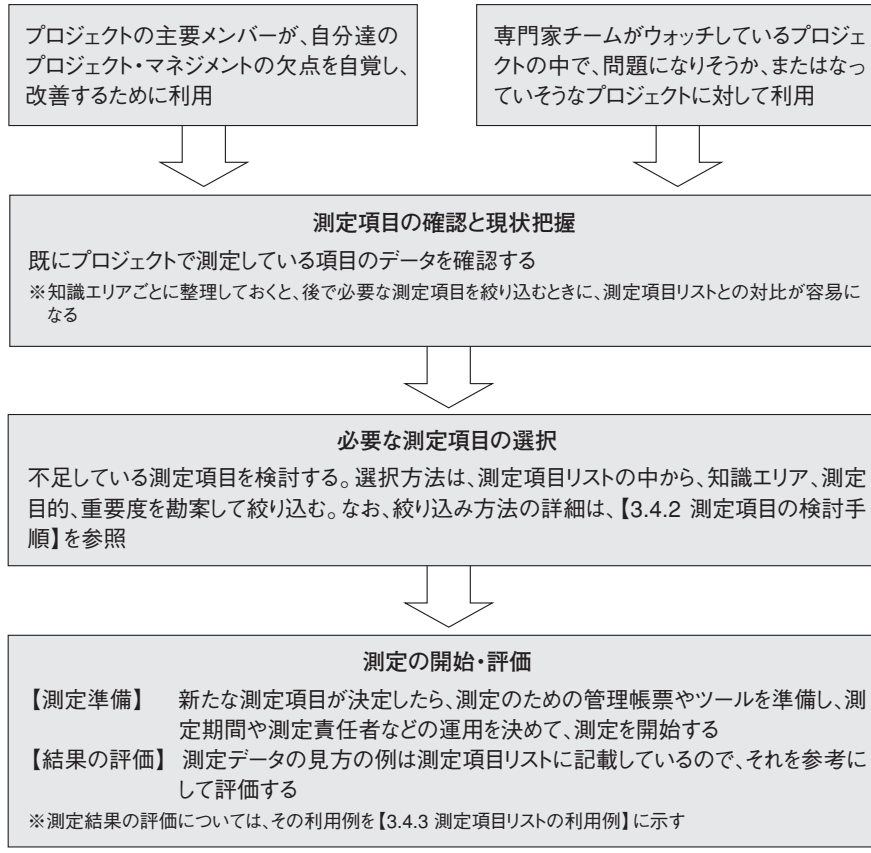
うに伝えるのが望ましいかを考慮する。

3.4 測定項目リストを用いた見える化

ここでは、「測定項目リスト」を用いて定量的に見える化する方法を説明する。測定項目リストは、プロジェクトを見える化するうえで、定量的な測定項目としてどのようなものがあるのかを一覧にしたものである（詳細は後述）。

図表3-23に、測定項目リストを用いた見える化の全体の流れを示す。まず、プロジェクト・マネージャやプロジ

図表3-23●測定項目リストを用いた見える化の全体の流れ



ェクトを支援するPMOのような専門家チームが、プロジェクトをどのように定量的に測定するか検討する。測定にはプロジェクト・メンバーへの負荷が高いものもあるため、プロジェクトの状況に応じて、測定を実施するかしないか、実施するならばその範囲や程度について検討する。

次の「測定項目の確認と現状把握」では、現在測定しているものを調査する。実際には、プロジェクト・メンバーには「測定している」という意識がなかったとしても、進捗報告書の内容やバグ管理表などの各種管理帳票を調べると、測定項目として捉えることのできる情報が含まれていることもある。後述する測定項目リストを参考にしながら洗い出してもよい。

「必要な測定項目の選択」では、現在測定している項目のうち、どの項目を徹底させるかを検討する。また、ほかにも必要な測定項目があれば、どのような項目を測定しなければならないかを検討する。

「測定の開始・評価」では、最初に測定準備を行う必要がある。管理帳票の準備や測定の仕組みを導入する。運用方法は、あらかじめ定めておかなければならない。

例えば、ソースコード管理システム（CVS）へのチェックイン、チェックアウトの回数を測定することに決めたとしても、開発者がローカルな環境で修正したものを月末に一括して反映させるという運用では、意味のある測定ができない。測定の目的を明確にし、運用ルールを決めた上で、プロジェクト・メンバーに徹底させる必要がある。こうした準備が完了したら測定を実施する。結果を評価し、どのような問題があるかを分析する。

3.4.1 測定項目リストと構成の説明

プロジェクトを定量的に見える化するためには、どのような測定項目が必要なのかを決定する必要がある。測定項目を定める際には、その測定目的を明確にし、測定にもれがないよう、網羅的かつ体系的な測定項目リストが必要である。

今回、ITプロジェクトの開発に長年携わってきた経験者の知見や世の参考文献を基に、「測定項目リスト」として体系化した。

本書では特に下流工程に焦点を当てて測定項目をリストアップした。また、これらの測定項目を整理するために、チェックシートに合わせて、知識エリアおよび拡張知識エリアで分類している。測定項目リストの全体は、【付録4 測定項目リスト】に掲載する。**図表3-24**はその概要である。

図表3-24●測定項目リストの概要

No	知識エリア	測定可能な概念/測定目的	測定データ項目
1	スコープ	機能的規模と変動（機能面でスコープが増加していないか）	機能変更要求数（未、済）
2			機能変更対応数（仕様変更対応数）
3			ドキュメントの規模（ページ数）
4			ソースコード行数
5			ソースコード変更行数
6	タイム	マイルストーンの達成状況の管理	結合テスト工程での作業単位の進捗
7			基本設計からプログラミング、単体テストまでの進捗
8		マスター・スケジュールの妥当性	マスター・スケジュールにおけるタスクのオーバーラップ度
9		クリティカルパス・スケジュールの達成状況の管理	結合テスト工程でのクリティカルパス作業の進捗
10		重要な機能の達成状況の管理	重要な機能数
11			重要な機能の作業進捗
12		作業進捗（テストの進捗）	テストケース消化数
13			テストチームごとのテストケース消化数
14		作業進捗（プログラム修正の進捗）	チェックイン、チェックアウト回数の推移（ソースコード更新頻度の推移）
15			ソースコード量の推移
16			ソースコード改変量の推移
17		作業進捗（ドキュメント修正の進捗）	ドキュメント量の推移
18			ドキュメント改変の推移
19		開発・テスト環境の充足度	開発環境の数
20			テスト環境の数
21	コスト	プロジェクト予算の管理状況	使った金額
22			アーンド・バリュー
23		機能追加時のコスト面での対応状況	追加請求額
24		プロジェクト予算の余裕度	残っている予算
25	品質	信頼性	検出バグ（現象）数
26			開発チームごとの検出バグ（現象）数
27			MTBF（平均故障間隔）
28			MTTR（平均修正日数）
29		信頼性（ドキュメントは修正されているか）	修正内容のドキュメントへの反映件数
30		信頼性（バグの傾向を分析しているか）	類似バグ調査実施回数
31			検出バグ数（現象）の分析実施回数

32	品質	信頼性（テスト計画書はレビューされているか）	テスト計画書のレビュー回数
33			テスト計画書のレビュー時の指摘数
34		信頼性（テストケースはレビューされているか）	テストケースのレビュー回数
35			テストケース・レビュー時指摘数
36		信頼性（テストデータはレビューされているか）	テストデータ・レビュー回数
37			テストデータ・レビュー時指摘数
38		信頼性（基本設計書はレビューされているか）	基本設計書レビュー回数
39			基本設計書レビュー時の指摘数
40		信頼性	コードクローン
41			データ項目の数
42			テストケース数
43			テストケース密度
44			手戻りテスト回数
45			プログラムごとのコーディング規約順守性
46			プログラムごとの単体テストケースの網羅率
47	人的資源	体制の強弱（経験の程度）	テスト工程に入っている基本設計者の人数
48			新人比率
49			プロマネの経験年数・経験プロジェクト数
50			対象業務経験者数
51			テスト経験者の数
52		体制の強弱（専任の程度）	メンバーのプロジェクト間の兼務率
53			メンバーのプロジェクト内兼務率
54			品質管理担当者の（プロジェクト内）専任者数
55			テスト・開発環境管理者の（プロジェクト内）専任者数
56			構成管理者（ライブラリアン）の（プロジェクト内）専任者数
57			リリース管理者の（プロジェクト内）専任者数
58		体制の強弱（技術レベル）	スキル評価
59	コミュニケーション	会議の出席状況	会議出席率
60		メール送受信状況	メール送受信数
61		協力会社との連携状況	チームごとの作業場所数
62	モチベーション	勤務状況	メンバーの労働時間（残業時間）
63	課題管理	作業進捗（課題が増加していないか）	課題数（未・済）
64	リスク	サービスインまでのリスクが増加していないか	リスク項目数

No	知識エリア	測定可能な概念/測定目的	測定データ項目
65	組織	協力会社体制の強弱	参画する会社数
66			階層数
67			地理的な開発拠点数
68		組織標準への準拠状況	組織標準の順守度合い
69		プロジェクト標準への準拠状況（標準に従って作業をしているか）	プロジェクトの標準の順守度合い
70		プロセスの標準への準拠状況（標準に沿った管理がされているか）	管理指標

3.4.2 測定項目の検討手順

何を測定するかは、プロジェクトを見える化するために重要である。しかし、測定のためには、時間、労力がかかるため、すべての測定項目を調べることはできない。ここでは、最低限必要な測定項目を選ぶために、測定項目の選定手順を示す。

（1）知識エリアによる絞り込み

下流工程では、本番稼働に耐え得る品質を確保できるかを把握するために、特にタイム、品質の2つの知識エリアに重点を置く必要がある。また、この時期にスコープが大幅に逸脱していると、タイム、品質に影響があるため、スコープにも重点を置く必要がある。

（2）測定可能な概念/測定目的による絞り込み

どのような測定を行うかを、その目的に応じて測定項目リストの「測定可能な概念/測定目的」を参照して絞り込む。

（3）重要度による絞り込み

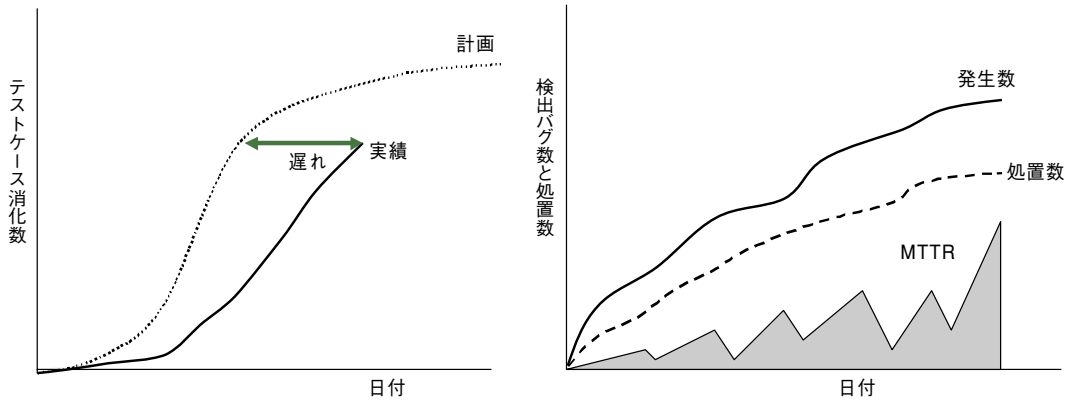
絞り込んだ「測定可能な概念/測定目的」の中から、重要度欄に★印が付いている測定項目に絞り込む。測定されていない項目があれば新たに測定する。

なお、測定項目を決定する際は、2つの点に改めて留意する必要がある。まず、測定には時間と労力がかかるため、費用対効果を勘案すること。次に、プロジェクトの特性に適した効果的な項目（必要最低限の項目）のみを測定することである。

3.4.3 測定項目リストの利用例

測定項目の見方や測定項目から分かることは、測定項目リストの「測定データの見方、分かること」欄に記載してい

図表3-25●テストスケジュール、検出バグの発生・処置の見える化例



る通りだが、複数の測定項目に関連したデータの見方の例を、下流工程であるテスト・スケジュールの進捗と検出バグの発生・処置状況から例示する。

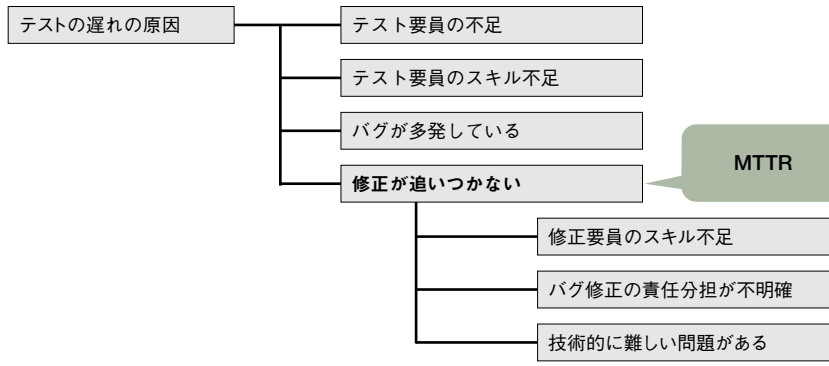
テスト・スケジュールの進捗とバグの発生状況などを見える化するには、毎日、テストケースの消化計画と実績の差異分析を行い、遅れが出ていないかどうかをチェックする必要がある（**図表3-25**）。これと合わせて、バグの発生・対応内容の把握を行う。特に留意すべきことは、個々のバグ修正までにかかった平均修正日数（MTTR：Mean Time To Repairに相当）の長さで、プロジェクトのパフォーマンスを評価することである。

例えば、テストに遅れが発生した場合には、**図表3-26**に示すような原因が考えられる。①テストを実施している要員のスキルが大丈夫か、②テスト項目の消化が進んでいないにもかかわらず、バグが多発していないか、③テストが遅れている原因が、バグのためにその先のテストに踏み込めないからではないか、などの点を考慮しながら測定データを見る必要がある。

3.4.4 プロジェクト・モニタリング・ツールEPMによる測定例

測定項目リストには多数の測定項目がリストアップされているが、このうちのいくつかは、先に紹介したEASEプロジェクトから提供されているEPMのようなプロジェクト・モニタリング・ツールで自動的に測定し、グラフ化して表示できる（詳細は【付録5 EPMツールの分析指標】を参照）。ここではそのEPMの出力例をいくつか紹介する。

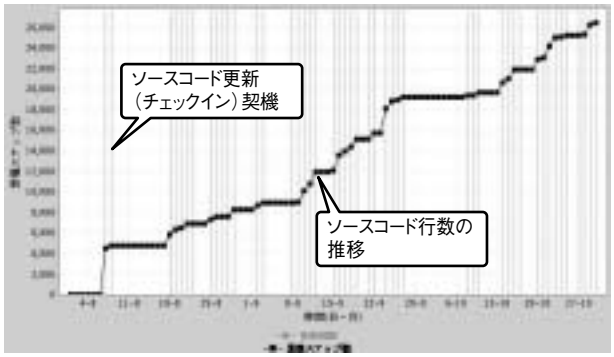
図表3-26●テスト遅延の原因



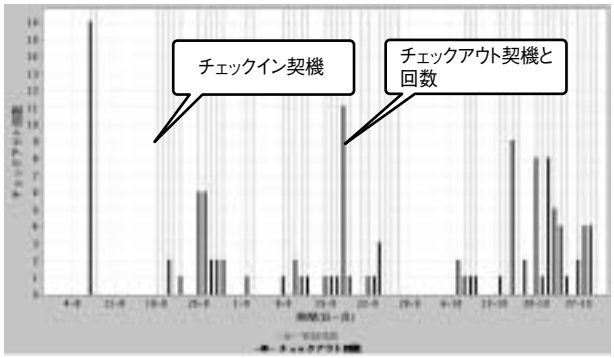
① テストを実施している要員のスキルが大丈夫か
② テスト項目の消化が進んでいないにもかかわらず、バグが多発していないか
③ テストが遅れている原因が、バグのためにその先のテストに踏み込めないからではないか
　③-1 修正要員のスキル不足はないか
　③-2 バグ修正の責任分担が不明確になってはいないか
　③-3 技術的な（上流工程や製品そのものに起因するような）難しい問題が発生していないか

【分析の観点】
・発生している問題点の事実を把握する（他人任せにせず、自分で見る）
・その問題点が起こっていることは、人に依るのか、環境に依るのか、スケジュールに依るのか、難易度に依って起こっているのかを分類する必要がある
▶人による場合
　スキル・要員数が関係する
　新人比率が高くはないか（特にバッチのテストは新人にやらせてはいけない）
▶環境による場合
　テスト環境がテストに十分な環境になっていない（性能、台数、ネットワークなど）
　テストに必要なデータが準備されていない
　プログラム構成管理の不足によるリリースミスが発生している
▶スケジュールによる場合
　元々計画していたスケジュールに根拠がない
　関連する他の作業の遅れに影響がないか（要員への多重作業の割当、共通プログラムの提供遅れなど）
▶難易度
　発生している問題の多くが、前工程に遡って設計し直す必要があるかどうか
　個々の修正の規模が大きくないか
　個々の修正の影響が広範にわたってはいないか

図表3-27●ソースコード更新（チェックイン契機）と行数の推移



図表3-28●チェックイン/チェックアウト回数の推移



（1）ソースコード行数の推移

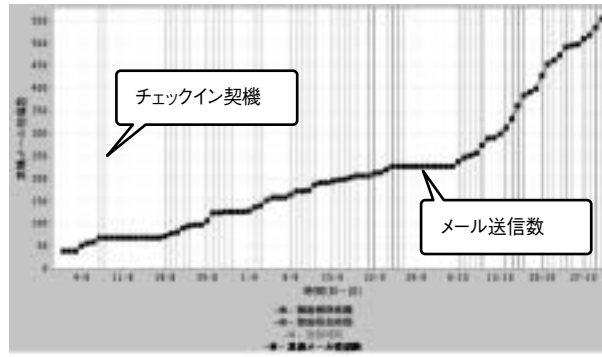
図表3-27は、ソースコード更新（チェックイン契機）とソースコード行数の推移を、時系列でグラフ化した例である。チェックイン契機は開発スタイルと開発作業の進捗に依存し、この例では、開発中は平日にほぼ毎日チェックインが行われ、時々休止の期間があったことが分かる。そして、これに合わせてソースコード行数が成長していく様子が示されている。またその傾きから、開発のペースが中盤以降で加速しているように見える。

図表3-28は、チェックイン契機とチェックアウトの契機および回数の推移を時系列でグラフ化した例である。チェックイン契機は図表3-27と同じである。このグラフから、チェックアウトは

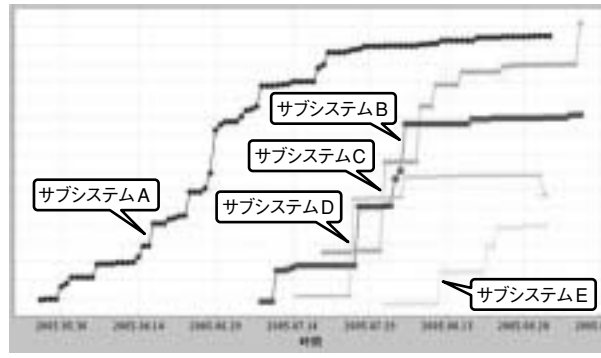
必ずしもチェックインと同じ頻度では行われず、またその回数もまちまちなことが分かる。影響度の大きい重要な修正と、それに伴うチェックインが行われたあとのチェックアウトの有無は重要で、対応したチェックアウトが無い場合、最新ファイルが周知されていない可能性がある。

図表3-29は、開発グループ内のメール送信数とチェックイン契機の推移をグラフ化した例である。チェックイン契機は図表3-27、図表3-28と同じである。この例では、EPMはチェックイン契機を「通常の変更時」、「障害発生時」、「障害修正時」で色分けして表示している。メール送信数の推移を見ると、障害修正（図では濃い色の縦線）が集中しているプロジェクトの後半に開発グ

図表3-29●メール送信数とチェックイン契機



図表3-30●異なる開発チームによる複数のサブシステムのソースコード行数推移の俯瞰



ループ内のコミュニケーションが活発になっていることが分かる。開発要員数は一定だったので、明らかに後半の開発作業が加速している様子がうかがえる。

図表3-27から図表3-29のグラフの横軸とチェックイン契機は揃えてあるので、この3つのグラフを鳥瞰することで、プロジェクトの開発作業の状況をさらに掘り下げて把握できるようになる。

図表3-30は、異なる開発チームが複数のサブシステムを開発しているプロジェクトで、各チームが開発したソースコード行数の推移を重ねて表示したものである。一見して、各開発チームの立ち上がりがばらばらなことが分かる。各チームの開発量の規模感や、開発ペースの違いなども分かる。後から開発を始めたチームが、先行したチームのソースコード行数を追い越すといった現象も起きている。

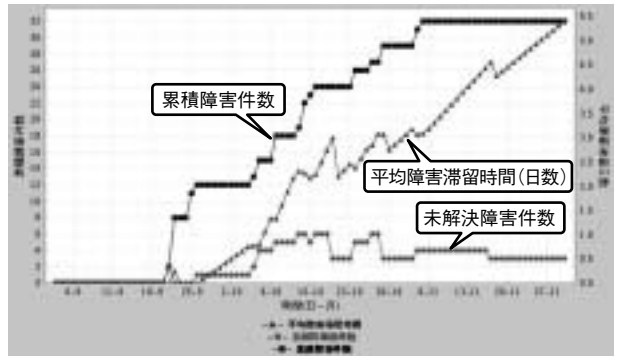
各チームの開発管理体制の一端をうかがい知ることもできる。ソースコード行数が細かく増加しているチームは開発現場で構成管理を実施していて、開発の実態を細かく把握していると想像できる。一方、大きな階段状に推移しているチームでは、ある時間単位でまとめた構成管理が行われ、管理の粒度が粗いと思われる。

こういった全体を俯瞰した定量的なプロジェクト把握は、複数社でサブシステムごとに開発を分担するマルチベンダー開発や、広域分散開発時のマネジメントに有効である。

(2) 検出バグ数の推移

図表3-31は検出バグ数の推移の例

図表3-31●検出バグ数の推移（累計障害件数、未解決障害件数、平均障害滞留時間（日数））

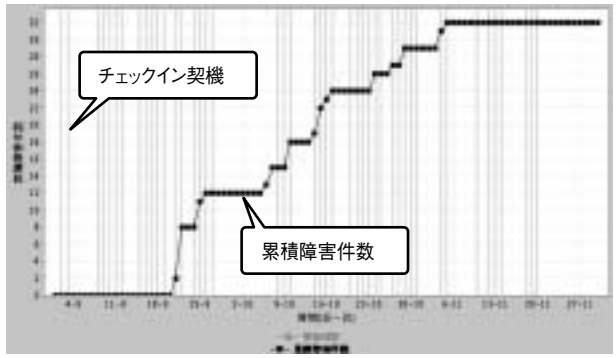


である。横軸を時間軸として、累積障害件数、未解決障害件数、平均障害滞留時間（日数）をグラフ化している。この例では、後半に新規障害の発生が止まって収束しているが、数件の未解決障害が残されたままで、平均障害滞留時間が増加し続けている。

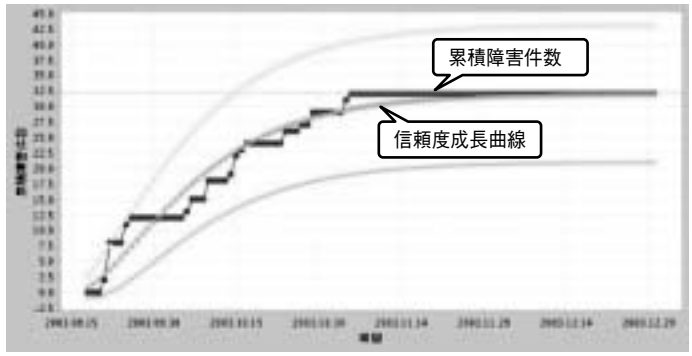
図表3-32は、上記の累積障害件数とチェックイン契機をグラフ化した例である。チェックインの休止期間と障害発生の時期的な関係などを読み取ることができる。

例えば、9月25日（25_9）から10月9日（9_10）の期間は、「累積障害件数」が増加していないことが分かる。しかし、この期間にチェックインも行われていない。つまり、累積障害件数がこの期間に増えていない理由は、単にチ

図表3-32●検出バグ数の推移（累計障害件数）とチェックイン契機



図表3-33●検出バグ（累計障害件数）と信頼度成長曲線



ェックインがされていない（開発作業が行われていない）からと推測することができる。

図表3-30と突き合わせてみても、この期間はメール送信がされていない。実際には開発担当者が休暇期間であったわけだが、このように累積障害件数とチェックインの状態を比べることで、プロジェクトの活動の様子を推測することができる。

また、この図の例では、チェックインとともに、障害件数が増加していることが分かる。このことから、チェックイン回数を障害発生件数の代替特性と捉えて、チェックイン契機をウォッチすることにより、障害発生の予測を行うことも考えられる（実際には、開発工程の進み具合や担当者の人数とスキルなど、様々な要素が絡み合うため、チェックイン契機で障害発生の予測に使うという考え方は一般的ではない）。

図表3-33は、この累積障害件数の推移に、信頼度成長曲線を重ねたものである。信頼度成長曲線は、過去の障害発生状況からその将来を予測するもので、障害発生の収束判断に用いることができる。EPMにはいくつかの計算式で信頼度成長曲線を表示する機能がある。

図表3-31から図表3-33は同じ累積障害件数のグラフ化の例で、これらを合わせて鳥瞰すれば、テスト工程のより深い判断に役立てることができる。開発チームの異なる複数のサブシステムのバグ検出状況をそれぞれグラフ化し、並べて鳥瞰してみてもいいだろう。バグの発生とそれらへの対処状況について、各開発チームの様々な状況を把握できる。先にも述べたように、マルチベンダー開発のマネジメントでは効果的である。

EPMは有用だが、これらの印刷されたグラフ類はいわば“止まった（静的な）情報”である。実際のEPMの出力はプロジェクト進行中に、リアルタイムに得られるダイナミックな情報である。図表3-27から図表3-33で示したようなグラフが時間軸とともに日々成長する形で提示されてゆく。これらを観測しながら、まだ誰にも見えないプロジェクトの将来を予測して的確な判断につなげていくことが重要である。

3.5 失敗事例データベースによる見える化

ここでは、SECが収集した77の失敗事例（以降「失敗事例データベース」と呼ぶ）を活用して、プロジェクトを見える化する方法を説明する。

プロジェクトの状況は、まずチェックシートによって定性的な状況判断を行

い、次にその状況を測定項目リストの指標を用いて定量的に把握する。さらに、過去の経験や知識を活用する方法として、対象とするプロジェクトの状況を失敗事例データベースと比較する方法がある。類似の状態や現象を見つけ出し、その進行過程や悪化状況を類推することができる。

ただし、現段階で失敗事例データベースは77事例であり、事例データベースとしてはまだまだ数が足りない。今後SECは、有識者の協力を得てより多くの失敗事例を収集する必要があると考えている。失敗事例データベースが500事例、あるいは1000事例となれば、プロジェクトで生じている状況とよく似た事例をデータベースから検索し、問題と対応策を見つけ出すことができるようになる。現在の失敗事例データベースは、そのための第一歩という位置づけである。

3.5.1 失敗事例データベースの構造

失敗事例データベースは、開発現場の有識者に記入してもらい収集した。有識者の選定は、プロジェクト・マネージャ経験10年以上、あるいは品質保証業務でプロジェクトの火消し経験10年以上を基準に行った。経験豊富な方々の貴重な体験が基になっているため、77事例といえどもデータベースの内容は現実的かつ迫力がある。

失敗事例データベースの一例を**図表3-34**に示す。各項目の説明は**図表3-35**の通りである。失敗事例データベースの全体は【付録3 プロジェクトにおける問題事象と対策事例】を参照してもらいたい。

具体的な失敗事例とその対応策をあらかじめ知識として身に付けておけば、プロジェクト遂行中に発生する様々な問題に対して適切な処置を実施することが可能になる。特に経験の少ないプロジェクト・マネージャにとっては有用な知識となる。

3.5.2 失敗事例データベースの整理

失敗事例データベースに登録されているプロジェクト失敗事例は、チェックシートや測定項目リストなどの他のツールと同様に15の知識エリアに分類して整理している。失敗事例の原因や発生個所を特定し、対応する知識エリアを割り当てるという方法で分類している。事象タイトルの内容を整理したものを**図表3-36**に示す。

この整理方法により、問題の予兆や発生を知識エリア単位に分けて考えることができ、速やかに類似事例にたどり着くことが可能となる。また、チェックシートや測定項目リストなどの他のツールとも連携しやすい。

図表3-34●失敗事例データベースのサンプル

スコープ

12 大幅な工数増加と工程遅延

顧客からの納期要求、追加・変更機能要求に対する反論ができず一方的な受入を余儀なくされ、工数増加と工程遅延を招いた

◆原因

仕様がはっきりしない状態で、過小見積もりしたため、プロマネ、管理要員、チームリーダーが必要数アサインされていなかった。そのため、十分な仕様変更管理ができなかった

◆対策

サブプロマネ投入。管理要員、チームリーダー追加投入。仕様の承認ルール、仕様追加・変更スキームと費用精算ルールなど、顧客と必要なルールの取り決め。障害管理、リリース管理、構成管理などプロジェクトチームが順守すべき規約・標準類を整備し、順守の徹底を図る

◆対症療法

・プロマネ支援の投入
・手順・ルールの確立

◆再発防止策（教訓）

・手順・ルールを事前に定義しておく
・顧客とのルール・手順の合意
・仕様確定の顧客との合意

図表3-35●失敗事例データベースの項目説明

項目		説明
1	No.	事例番号
2	知識エリア	PMBOKの9つの知識エリアに、6つの拡張知識エリアを加えた15 の知識エリアを示している。詳細は「付録 7 知識エリアについて」を参照
3	事象タイトル	問題事象を示すキーワード
4	事象	プロジェクト遂行中に発生した問題について記述している
5	原因	問題が発生した原因について記述している
6	対策	そのプロジェクトでとった対策を記述している
7	対症療法	問題が発生したときに速やかに行うべき処理について記述している
8	再発防止策（教訓）	事例と同じ問題を繰り返さないために実施すべき処置を記述している

図表3-36●失敗事例データベースおける事象タイトルの一覧

知識エリア	問題の発生個所	事象タイトル
統合	変更管理 計画の策定	•中核SEの離脱 •仕様凍結の遅延 •特定機能の進捗遅れ、品質劣化 •検収時期に一括納品できず •パッケージとカスタマイズのソースコード範囲が不明確
スコープ	スコープ定義 スコープ検証 スケジュール管理	•テスト工程で設計仕様変更が多発 •開発側の運用イメージ不足と仕様説明不足 •顧客の機能要件説明能力不足と開発側の顧客ニーズ把握ミス •業務設計の中心となる設計リーダーが倒れて復帰できない •大幅な工数増加による採算悪化 •大幅な工数増加と工程遅延 •納期に間に合わない •仕様があいまいなままシステムを提供 •プロジェクト・マネージャが多忙 •共同受注案件での開発分担不明確による混乱 •要件定義が不十分で、テスト工程になってリリース必須機能の漏れが発覚 •オープンシステムのプロジェクト経験がない •要求仕様が固まらないまま開発に突入
タイム	アクティビティ定義 スケジュール管理	•テストの進め方が分からず、テスト計画もできない •稼働1カ月半前でも結合・総合テスト計画が不十分 •大幅な工数増加による採算悪化 •共通ライブラリのスケジュール遅れ、品質不良 •プロジェクト全体に疲弊感が漂っている
コスト	コスト見積もり	•見積もり時の前提想定不足 •テスト工数を小さく見積もってしまった
品質	品質計画 品質管理	•メーカー提供プロダクトの不具合でシステムが動かない •システムの品質が悪い •大量印刷時間が長い •ユーザー研修で急に処理が遅くなる •発注時の丸投げによって失敗 •パッケージのバグフィックスに時間がかかる •大量データ処理方式の設計ミス •あいまいな仕様のまま実装し、テスト工程で要件不一致が多発 •外字印字の不良 •運用中の業務停止 •プログラムの品質が悪い •パフォーマンス設計の不備

人的資源	組織計画 チーム育成 要員調達	•指揮命令系統が不明確 •プロジェクト編成の中でプロジェクト・マネージャ的な人材が2人存在する形になり混乱 •他業種から転進した元請け会社が未熟で混乱 •DBMS の技術的な問題で対応できない •結合テストで品質不良が発生 •受注後の要件詳細化、基本設計体制に不備 •大幅な工数増加と工程遅延 •業務担当リーダーの離脱 •協力会社のリーダー、中核要員がくるくる変わる •要員のスキルが低すぎてまったく役に立たない •個別チームの状況を把握していないため、総合テストに入っていいのか分からない •キーパーソンが疲弊している •協力会社が多階層になり責任範囲があいまいに •実質的なマネジメントを協力会社の開発グループが実施 •総合テストの進捗が悪い •テスト作業に無駄が多い •サービス開始の可否が判断できない •総合線表がない •顧客側と開発側のプロジェクトの一体感欠如
コミュニケーション	コミュニケーション計画 実績報告 情報配布	•プロジェクト・マネージャ、リーダーと実務担当者の関係が悪化してコミュニケーションも作業も進まない •役割分担が不明確 •複雑な開発組織内で口頭による依頼、周知が多用され混乱 •事務方との連携が悪く、開発グループが客先で孤立 •総合テストが品質不良で進まない •仕様書と実装の差異に顧客がクレーム •開発チームへの指示系統が複数
リスク	リスク識別	•顧客側のプロジェクト・マネージャが交代してしまった •協力会社が設計を完了した時点で次工程の開発を断ってきた •業務有識者不在での総合テスト実施
調達	調達計画 契約管理	•契約前に作業完了 •ベンダー提供の業務パッケージの品質不良による運用の中止 •営業時に営業SEが決めたブラウザが実用に耐えなかった •営業時に営業SEが決めたDBMSの最新版を自社ミドルウェア製品がサポートしていなかった •パッケージ利用によりブラックボックス化が問題に •外部調達のパッケージ製品の品質が悪くて使えない
基本動作 （拡張）		•結合テストで重複障害やデグレートが多発
モチベーション （拡張）		•自社のプロジェクト・マネージャが体調不良で倒れた •プロジェクト・マネージャのストレスとの戦い
顧客（拡張）		•顧客側の担当者が能力・経験不足でまったく役に立たない

失敗事例データベースとしてはまだまだ事例件数が少ないが、個々の事例の内容は具体的で示唆に富むものとなっている。SECは今後も収集活動を続けて、事例件数を増やし、実用的なデータベースを公開、普及していくことが重要だと考えている。

第4章

言える化

4.1 概要

プロジェクトはその性質上、何もコントロールせずにいると、必ずと言っていいほど問題が発生する。例えば、利用しようとしているパッケージの品質や機能の問題、顧客とベンダーのコミュニケーションの問題、メンバーのやる気、モチベーションに関する問題など、実際のプロジェクトの現場では実に様々な問題が発生する。

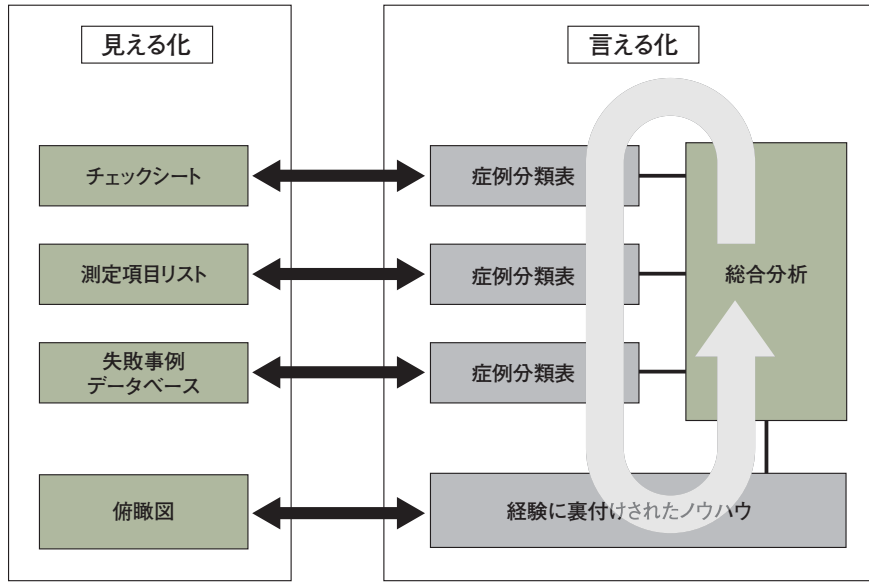
いつ発生してもおかしくないプロジェクトの問題に対して、プロジェクト・メンバーは「問題がどこに潜んでいるのか」を常に意識し、まだ潜伏している問題が大きくならないうちに対策を打つことが大切になってくる。とはいえ、どのような問題が発生するかは、経験を積んだプロジェクト・マネージャでなければなかなか気が付かないというのが実情である。そこで、第3章では俯瞰図、チェックシート、測定項目リスト、失敗事例データベースを用いた「見える化」について説明した。

しかし、複合的な問題は、単独のツールを用いた「見える化」手法によってだけでは症例を特定できない。症例を特定するには、**図表4-1**に示すように複数のツールを用いて、経験に裏付けされたノウハウを活かし、何が問題と「言える」かを総合的に分析することが必要になる。第4章では、プロジェクトの見える化手法のうち、「言える化」により問題を特定する手法を紹介する。

プロジェクト活動で目の当たりにする問題は、人間の病気に例えることができる。問診票で分かるような問題もあれば、レントゲンを撮影したり精密検査を受けたりしないと分からない問題もある。このようにプロジェクトの問題は、医療における病気と捉えると分かりやすい（**図表4-2**）。

プロジェクトで発生する問題は個々のプロジェクトの事情によってそれぞれ異なるが、過去のプロジェクトの失敗事例（症例）を集めて調査すると、その発生パターンを分類できる。また、プロジェ

図表4-1●プロジェクトの見える化手法における「言える化」の位置付け



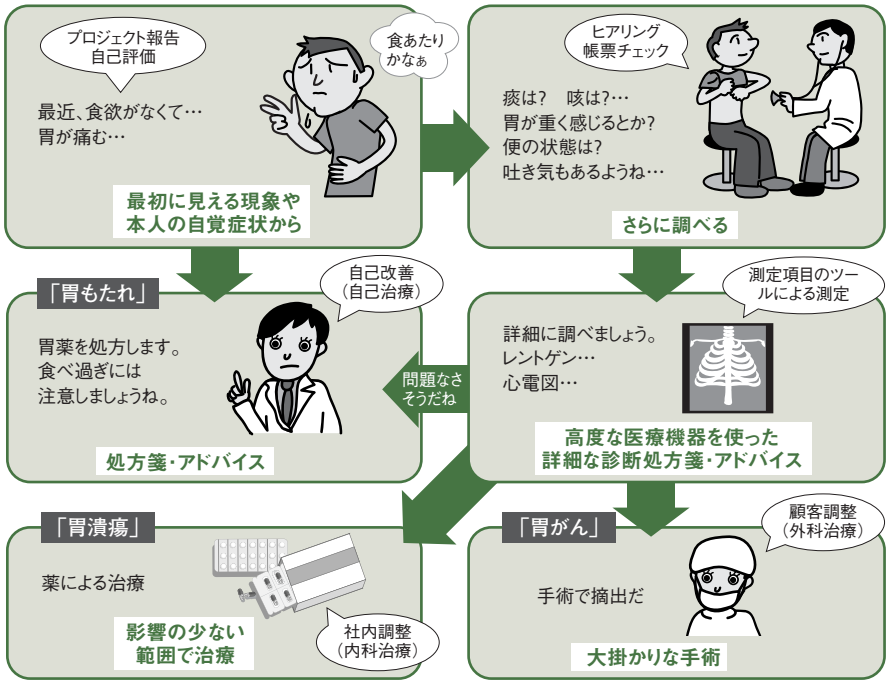
クトで発生した問題に対する対応策についても、程度の差はあるにしても、いくつかの方法に集約できる。プロジェクトを実際に遂行する立場にいる者にとって、この問題の発生パターンを理解しておくことは、実際に発生した問題への対処方法を考える上での参考になる。さらに、まだ発生していないが、これから発生し得る問題の発見という観点で、気付きが得られる。

プロジェクトの問題は、実に様々な原因から発生し、様々な症状となって現れる。一人の人間が、経験や勘に頼って病状を判断し、処方することは得策ではない。原因だと思っていたものが、実は本当の原因ではなかったり、もっと大きな原因を見逃してしまったりする危険性もあるからだ。

4.2 統合的アプローチによる「言える化」の流れ

プロジェクトの問題は症状として現れてくる。第3章で解説した「見える化」の診断や分析では、単独の原因で現れている問題を把握することはできる。だが、複数の原因が絡む複合問題

図表4-2●プロジェクトの問題は医療の観点で捉えると理解しやす



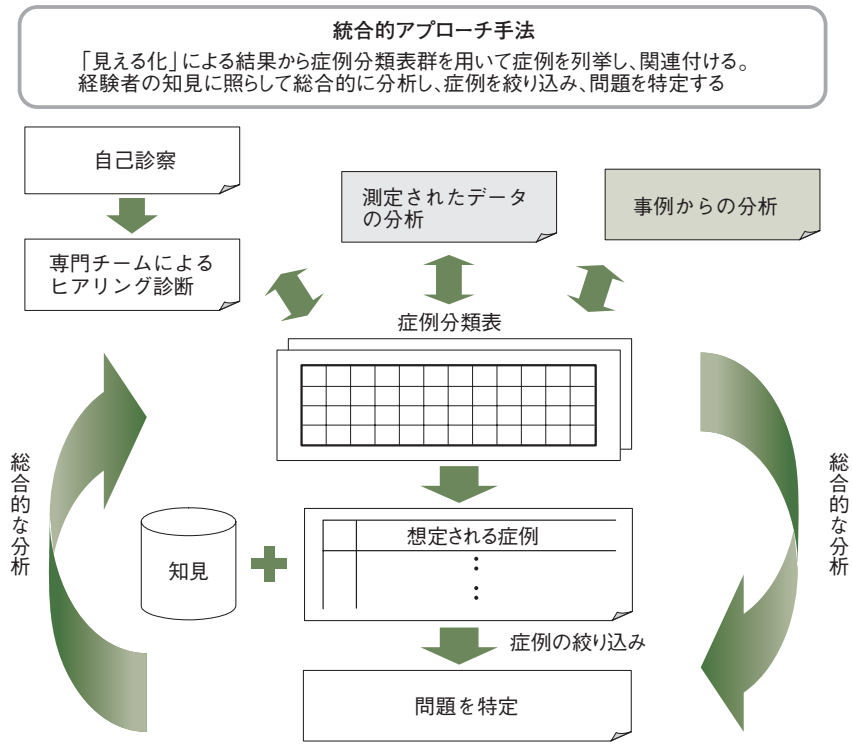
の場合には、ここで述べる統合的アプローチ手法を用いないと、問題の特定と解決策を見出すことができない。

統合的アプローチ手法とは、**図表4-3**に示す通り、第3章で説明したチェックシート、測定項目リスト、失敗事例データベースを、「症例分類表」（詳細は後述）を使って統合的に活用する手法である。このように、複数の「見える化」の結果を、総合的に整理し、さらに経験者の知見により問題を絞り込むアプローチは、これまでにない試みである。

4.3 統合的アプローチ手法で用いる症例分類表群の構成

医療の例で考えると、身体に異常（問題）が生じた場合、医師は「胃に潰瘍がある」、「血液中の血糖値が高い」、「大腿骨にヒビが入っている」のように、

図表4-3●統合的アプローチ手法による「言える化」の流れ



その問題となる現象がどの部位に発生しているのかを特定する。これをITプロジェクトの場合で考えると、「コミュニケーションが不足している」、「キーパーソンに負荷が集中しすぎている」、「テストの実施方法に不備がある」といった特定の仕方になるはずだ。

医療とITプロジェクトのいずれの場合でも、「問題発生個所」と「状態」に注目して診断している。問題発生個所は「コミュニケーション」や「キーパーソン」、「テストの実施方法」であり、それぞれ「不足している」、「負荷が集中している」、「不備がある」といった状態を呈している。

この「問題発生個所」と「状態」に注目し、過去のプロジェクト失敗事例を分析して体系化したものが「症例分類表」である。症例分類表では、ITプロジェクトにおける問題発生個所とそ

の状態を縦軸と横軸に取り、それぞれに対応する症例を分類したマトリックス表である。

ここでは、この症例分類表の構成とその考え方について説明する。

4.3.1 症例分類表の見方

症例分類表は4種類ある。まず、縦軸の「問題発生個所」と横軸の「状態」の組み合わせから、どのような症例パターンがあるのかを一覧したのが「症例分類表（一般マップ）」である。縦軸と横軸の構造は同じだが、中身を第3章で説明したヒアリングシート、測定項目リスト、失敗事例データベースの項目とマッピングさせたものを、それぞれ「症例分類表（ヒアリングマップ）」、「症例分類表（測定項目マップ）」、「症例分類表（事例マップ）」と呼ぶ。「症例分類表（ヒアリングマップ）」はヒアリングシートのチェック項目番号とリンクしている。「症例分類表（測定項目マップ）」は、測定項目リストの測定項目番号と関連付けている。「症例分類表（事例マップ）」は、事例番号と結び付けている。これらをまとめて「症例分類表群」と呼ぶことにする。

この症例分類表を用いることで、3つの見える化ツールを関連付けることができる。もし新たにプロジェクトを測定するためのツールを見える化の手法として取り込む場合でも、この症例分類表とリンクを張ることにより、これまで説明してきたヒアリングシート、測定項目リスト、失敗事例データベースと関連付けられる。

より具体的なイメージとして**図表4-4**に、ヒアリングシートと症例分類表（ヒアリングマップ）の関連の例を示す。例えばヒアリングシートのチェック項目番号H1に問題ありと感じたら、症例分類表（ヒアリングシート）の中から「H1」を含むセルをすべて探し出し、その縦軸と横軸を調べれば症例が分かる仕組みになっている。

なお、症例分類表群の全体については、【付録6 症例分類表】を参照してもらいたい。

4.3.2 症例分類表の構成

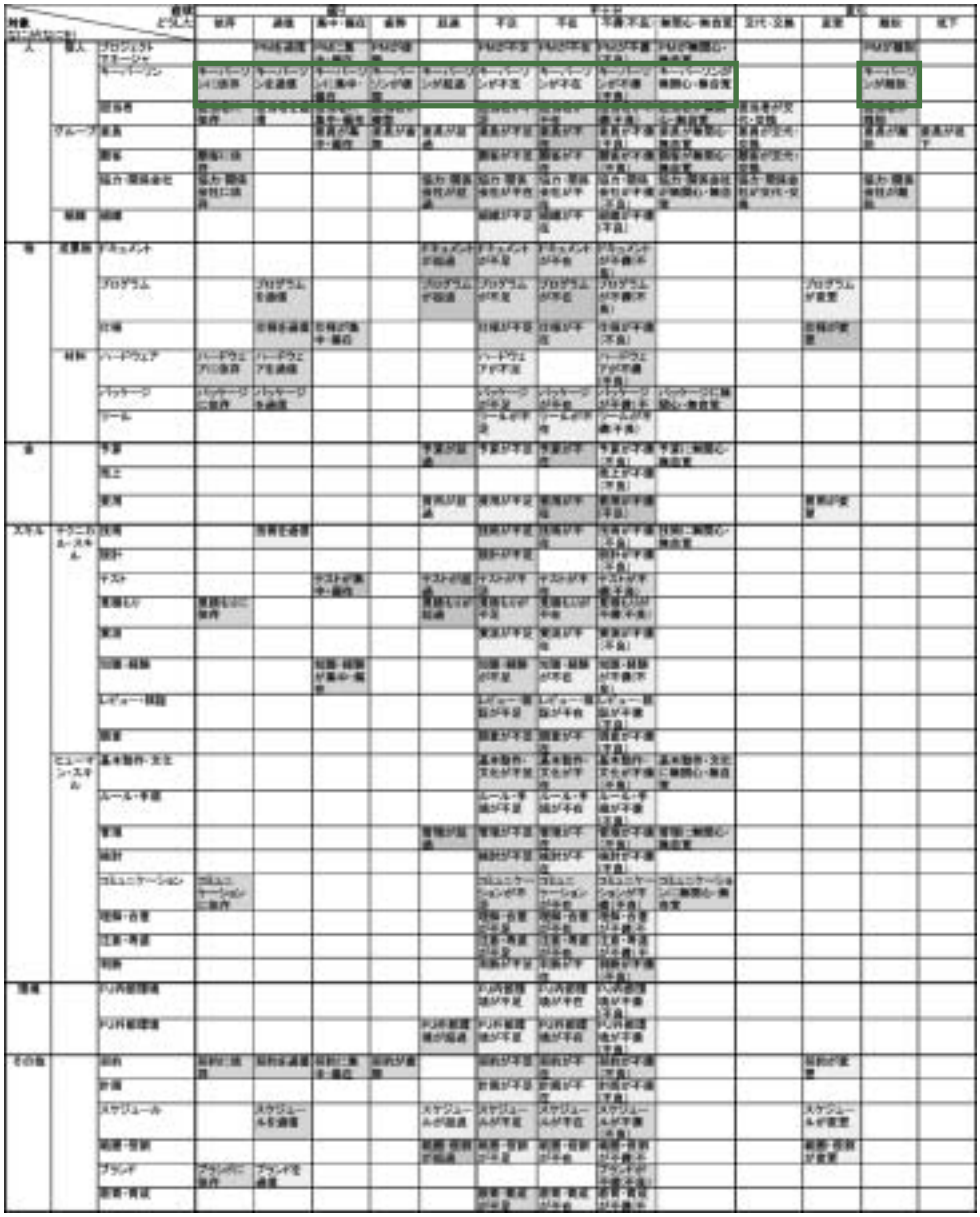
症例分類表の縦軸の「何が（何を）」は、プロジェクトで管理する対象（問題発生個所）を示している。横軸の「どうなっている（どうした）」は、その対象の症状を示している。

ここで**図表4-5**の症例分類表（一般マップ）を見てもらいたい。これはプロジェクトで起こり得る症例パターンの一覧である。例えば、対象の「キーパーソン」に関係する症状には、「依存」、「過信」、「集中・偏在」、「疲弊」、「超過」、「不足」、「不在」、「不備（不良）」、「無関

図表4-4●ヒアリングシートの症例分類

ヒアリングシート No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領
H1	統合	計画のオーソライズが承認され、宣言されているか？	計画を説明するにあたって、その計画が顧客や関連部門及び、上位部門との間できちんと承認されているかどうかを明確にしていること。 例えば、以下のような観点で確認を行なう。 ・キックオフ会議が開催されているか？ ・キックオフ会議に参加していたメンバーには開発者メンバーや上層部は入っていたか？ ※途中から参加したメンバーにはプロジェクト計画について説明をきちんと実施しているか？	今からやっても遅いかも知れないが、今からやる意味もある
	統合	重要度の高いミッションクリティカルな機能に対して、適切なテスト計画が策定されているか？	テスト計画書にミッションクリティカルな機能のテスト計画が[illegible]	(1)ミッションクリティカル性の高い機能要件は失敗してはならない機能なので、計画-収計時から入念に管理され、実行され、監視される必要がある。 (2)特にテスト内容が不十分の場合、業務の引渡し時や運用開始時に問題が発見された場合は大きな問題とな

図表4-5●症例分類表（一般マップ）



心・無自覚」、「離脱」がある。それぞれの症状に助詞を付け、「キーパーソンに依存」、「キーパーソンを過信」、「キーパーソンに集中・偏在」、「キーパーソンが疲弊」、「キーパーソンが超過」、「キーパーソンが不足」、「キーパーソンが不在」、「キーパーソンが不備（不良）」、「キーパーソンが無関心・無自覚」、「キーパーソンが離脱」と表現するとプロジェクトの症状が分かるようになっている。

(1) 対象（縦軸）の表現

症例分類表の縦軸は、プロジェクトの管理対象を表す。過去のプロジェクト失敗事例を基に、どのような部位に問題が発生するかを整理したものであり、全部で40項目ある。

プロジェクトを一つの事業体と捉え、管理の対象を経営資源としての「人」、「物」、「金」で分類するとともに、ITプロジェクトにとって重要な要素である「スキル」、「環境」なども分類項目として加えた（**図表4-6**）。

ここで定義している「対象」（プロジェクトの管理対象項目）は、プロジェクト・マネジメント経験者の知見や実例を基に定義した分類体系になっているため、漏れやダブりのない完全な体系ではない。今後、様々な事例を集計・分析し、新たなプロジェクトの問題発生パターンが現れた場合には、この「対象」を見直す必要があるだろう。

(2) 症状（横軸）の表現

症例分類表の横軸は、対象が「どうなっている（どうした）」という状態を示しており、全部で13項目ある。実際に経験したプロジェクトの事例から得られる症例、ヒアリングシートや測定項目から推察される症例を検討することにより定義した。これらの状態は「偏り」、「不足」、「変化」という3つのカテゴリで分類している（**図表4-7**）。

例えば、プロジェクトの管理対象である「人・個人」をプロジェクト・マネジメントの観点から考えると、設計の詳細化やプログラミングに向けては、担当者が「不足」するという状態が起こりやすく、設計から従事した要員や経験のある要員に対して、キーパーソンに「依存」するという状況が生じ得る。さらに、特定の個人への依存が続くとキーパーソンが「疲弊」し、その結果、キーパーソンの「離脱」というプロジェクトに大きな変化（衝撃）を与えることになる。

4.4 統合的アプローチ手法の活用方法

症例分類表群を用いて、プロジェクトの問題を特定する（言える化）手法

図表4-6●症例分類表における「対象」

対象 何が（何を）			説明
人	個人	プロジェクト・マネージャ	人は、個人とグループと組織の3つのサブグループ構成とし、個人はプロジェクト・マネージャ、キーパーソン（プロジェクト内で業務、技術、チームの中核メンバー）、担当者（プロジェクトに参加する人）を指す。グループはプロジェクトの関係者の集合体で、要員、顧客、協力・関係会社を指す。組織はプロジェクト内外にある目的や機能を担った部門・部署を指す
		キーパーソン	
		担当者	
	グループ	要員	
		顧客	
		協力・関係会社	
	組織	組織	
物	成果物	ドキュメント	物は、成果物と材料の2つのサブグループ構成とし、成果物は計画書、設計書、テスト項目表、議事録、報告書などのプロジェクトで作成するドキュメント、プログラムと仕様（プロジェクトで作成する対象を定義するもの）を指す。材料は、プロジェクトで利用するハードウェア、パッケージ・ソフトウェアとツールを指す
		プログラム	
		仕様	
	材料	ハードウェア	
		パッケージ	
		ツール	
金		予算	金は、予算（プロジェクトで利用できる経費）、売上（プロジェクトに対して顧客から得られる対価）、費用（プロジェクトが使った経費）で構成した
		売上	
		費用	
スキル	テクニカル・スキル	技術	スキルは、技術、基本動作・文化の2つのサブグループ構成とし、技術は、設計、テスト、見積もり、実測（プロジェクトで行った結果を定量/定性的に示すこと）、知識・経験、レビュー・検証、調査を指し、基本動作・文化は、基本動作・文化、ルール・手順、管理（PDCAのサイクル）、検討、コミュニケーション（情報や意志の疎通）、理解・合意、注意・考慮、判断を指している
		設計	
		テスト	
		見積もり	
		実測	
		知識・経験	
		レビュー・検証	
		調査	
	ヒューマン・スキル	基本動作・文化	
		ルール・手順	
		管理	
		検討	
		コミュニケーション	
		理解・合意	
		注意・考慮	
		判断	
環境		プロジェクト内部環境	環境は、プロジェクト内部環境（プロジェクトでコントロールできる環境を指し、例えば作業環境、テスト環境、開発環境がある）とプロジェクト外部環境（プロジェクトでコントロールできない環境を指し、法制度、会社財政などがある）で構成した
		プロジェクト外部環境	
その他		契約	その他には、計画、ブランド、教育・育成がある。計画は、契約（顧客、協力・関係会社などのプロジェクト責任の一端を担う会社との約束事）、計画（プロジェクト計画）、スケジュール（プロジェクト計画を詳細化し実行ベースで進捗などを表せるもの）と範囲・役割（プロジェクトが対象とする範囲や個人やグループが担当する役割）を指している。ブランドとは、「ブランドに起因したもの」を示している。教育・育成とは、プロジェクトの各種作業を通じて行うOJT（オン・ザ・ジョブ・トレーニング）である育成と自己啓発を目的とした教育などを指す
		計画	
		スケジュール	
		範囲・役割	
		ブランド	
		教育・育成	

図表4-7●症例分類表における「症状」

症状 どうした	偏り					不十分				変化			
	依存	過信	集中・偏在	疲弊	超過	不足	不在	不備（不良）	無関心・無自覚	交代・交換	変更	離脱	低下
説明	依存、過信、集中・偏在、疲弊、超過の総称で、プロジェクトのリリースに「偏り」が生じた状態を示す					不足、不在、不備（不良）、無関心・無自覚の総称で、プロジェクトに必要なものが「不十分な」状態を示す				交代・交換、変更、離脱、低下の総称で、プロジェクトの遂行によって「変化」する状態を示す			

を紹介する。症例パターンを絞り込めたら、さらに経験者の知見を用いた総合的な分析を行い、問題を特定して、改善を行う。

ここでは、症例分類表群を用いた4つの統合的アプローチ手法を紹介する。

4.4.1 活用方法①「ヒアリングシートから症例分類表を介し、測定項目を調べる」

PMOのような専門家チームがプロジェクトの状況を把握する場合、最初にヒアリングシートを基にしてプロジェクト・マネージャにヒアリングを実施することが多い。この場合、ヒアリングした結果は、あくまでもプロジェクト・マネージャの意見や、表面的な現象から分かる状況を確認したものなので、さらに踏み込んだ調査を行うのが一般的である。見識の深い専門家であれば、ヒアリングの内容からどの部分を踏み込んで調査すべきかを考えることができるだろう。

しかし、そうでない場合、調査が場当たり的になることも少なくない。ここでは、ヒアリングシートを使って問題個所が見えてきたときに、その問題を測定するための測定項目を、測定項目リストから探し出す方法について説明する。

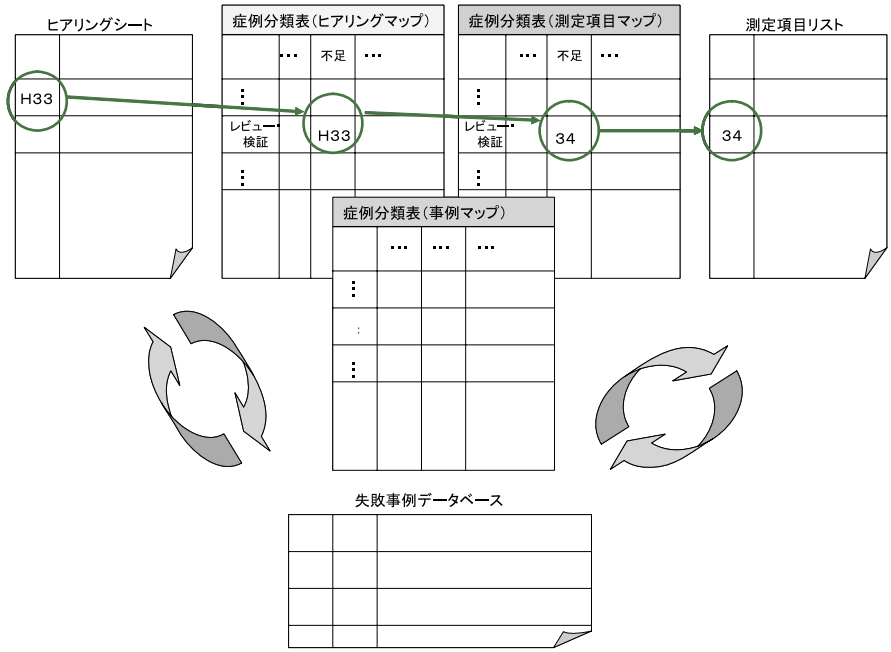
専門家によるヒアリングで何か問題の兆候に気付いた場合、症例分類表（ヒアリングマップ）を参照して症例を求める。そして、その症例に該当する測定項目を症例分類表（測定項目マップ）により求めればよい。これにより、定性的アプローチであるヒアリングの結果を、定量的な視点で具体的に分析できるようになる。

ヒアリングシートの分析で問題の可能性が認識されたら、次の手順で症例を識別する（**図表4-8**）。ここでは、ヒアリングシートの「テスト（連結/総合）計画とそのレビュー結果は問題がないか？」（チェック項目番号H33）という点に専門家チームが問題の予兆を感じ取ったと仮定しよう。

(1) ヒアリングシートの当該チェック項目番号H33が、症例分類表（ヒアリングマップ）のどこのセルにあるかを探す。症例分類表（ヒアリングマップ）のイメージは図表4-4を参照してもらいたい。

(2) 症例分類表（ヒアリングマップ）にて該当の「H33」を見つけたら、そのセルが属する行の左端（「テスト」、「レビュー・検証」、「計画」）に注目する。これによって問題のカテゴリ、即ち問題の対象（「何が(何を)」）を知ることができる。

図表4-8●症例分類表の活用方法①



⑶ 次に、そのセルが属する列の最上段（「不足」、「不在」、「不備（不良）」）に注目する。これによって問題の状態、即ち問題の症状（「どうした」に相当）を知ることができる。

⑷ 上記の問題の対象と問題の症状を組み合わせることにより、症例パターン（つまり、「何が」「どうした」）を識別できる。症例分類表（ヒアリングマップ）から、症例として「テストの不足」、「テストの不備（不良）」、「レビュー・検証の不足」、「レビュー・検証の不在」、「レビュー・検証の不備（不良）」、「計画の不足」、「計画の不備（不良）」が分かる。

⑸ 最後に、症例分類表（測定項目マップ）を参照する（この表の詳細は【付録6 症例分類表】を参照）。まず、症例分類表（ヒアリングマップ）の「レビュー・検証の不足」

と同じセルに記載されている症例分類表（測定項目マップ）の中身を確認する。症例分類表（測定項目マップ）の「レビュー・検証の不足」のセルには「34」と記されている。測定項目リストで「34」の測定項目（テストケースのレビュー回数）を特定し、そこに記載された測定方法（テストケースのレビュー記録）により詳細を確認する。

なお、この例でも分かる通り、ヒアリングシートの一つのチェック項目番号は、通常、症例分類表（ヒアリングマップ）の複数個所に存在する。これは一つのヒアリング項目が複数の症例に該当することを示唆している。

これを病気に当てはめてみると分かりやすいだろう。例えば、「鼻水がでる」（チェックリストの一項目に相当）という事象は風邪、鼻炎、花粉症（それぞれが1つのセル＝症例パターンに相当）などの可能性があることをイメージすればよい。

4.4.2 活用方法②「測定項目リストから症例分類表を介し、ヒアリング項目を調べる」

2つ目の活用方法は、活用方法①の逆パターンである。例えば、プロジェクト側で何らかの測定をすでに行っていて、その測定結果を専門家チームが見た場合、その測定結果を基に現象を説明できるだろう。しかし、ヒアリングシートとの関連を調べれば、新たにヒアリングすべき質問を探すことができ、ほかの視点からも調査できることに気付く可能性がある。

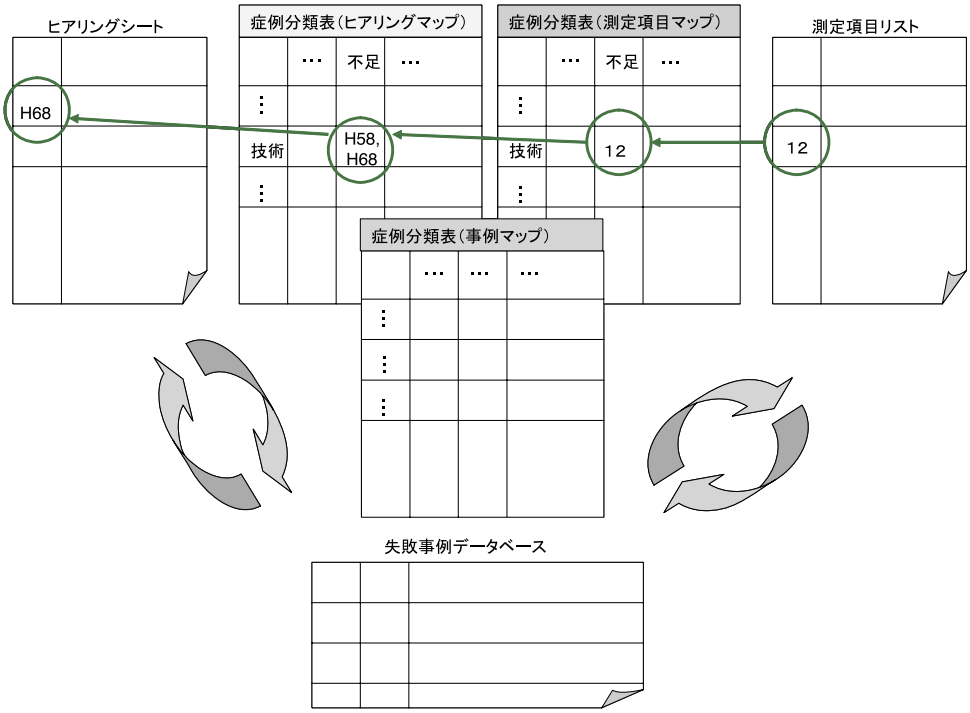
この方法は、プロジェクトで常に測定値をモニタリングしておいて、測定しているデータがしきい値を超えたら（予想しない値を示したら）、速やかに実施するとよい。症例分類表（測定項目マップ）と症例分類表（ヒアリングマップ）によって、定量的な測定結果を定性的な視点で診断し、数値だけでは判別できない状況を察知できる。

例えば、測定項目リスト「12」で測定しているテストケース消化数が計画と異なることが認識されたら、次の手順で症例を識別し、ヒアンリングでさらに何を確かめるべきかを調べてみよう（**図表4-9**）。

(1) 測定項目リストの当該項目の「12」が、症例分類表（測定項目マップ）のどこのセルにあるかを探す。

(2) 症例分類表（測定項目マップ）にて該当の「12」を見つけたら、そのセルが属する行の左端（「技術」、「実測」、

図表4-9●症例分類表の活用方法②



「知識・経験」）に注目する。これによって問題のカテゴリ、即ち問題の対象（「何が（何を）」）を知ることができる。

(3) 次に、そのセルが属する列の最上段（「不足」、「不在」、「不備（不良）」）に注目する。これによって問題の状態、即ち問題の症状（「どうした」に相当）を知ることができる。

(4) 上記の問題の対象と問題の症状を組み合わせることにより、症例パターンを識別することができる。症例分類表（測定項目マップ）から症例として「技術が不足」、「技術が不備（不良）」、「実測が不足」、「実測が不在」、「実測が不備（不良）」、「知識・経験が不足」、「知識・経験が不備（不良）」が分かる。

⑸ 最後に、症例分類表（ヒアリングマップ）を参照する。症例分類表（測定項目マップ）で分かった症例の一つ「技術が不足」に対応する症例分類表（ヒアリングマップ）のセルから、ヒアリングシートのチェック項目番号（H58、H68）を見つけ出す。この番号を基に、ヒアリングシートの該当チェック項目のヒアリングを実施する。

4.4.3 活用方法③「ヒアリングで判明した課題に対し、症例分類表、失敗事例DBから対策を調べる」

ヒアリングを実施してプロジェクトに何らかの問題を見つけたときに、その問題に類似する過去の失敗事例を探し、それを参考にして今後起こり得る他の問題や対処方法などを推測することもできる。本来ならヒアリングだけではなく、定量的な測定を使って事実確認をするべきではあるが、プロジェクトで発生する問題は必ずしも定量化できるものばかりではない。例えばパッケージ製品に関するトラブルや、顧客とのコミュニケーション不足の問題など、失敗事例を調べるほうがより分かりやすい対応になることが多い。

プロジェクトのヒアリングを通じて課題が明らかになった場合、症例分類表（ヒアリングマップ）から症例を識別し、症例分類表（事例マップ）の症例からその対策例（対症療法や再発防止策）を見けることができる。

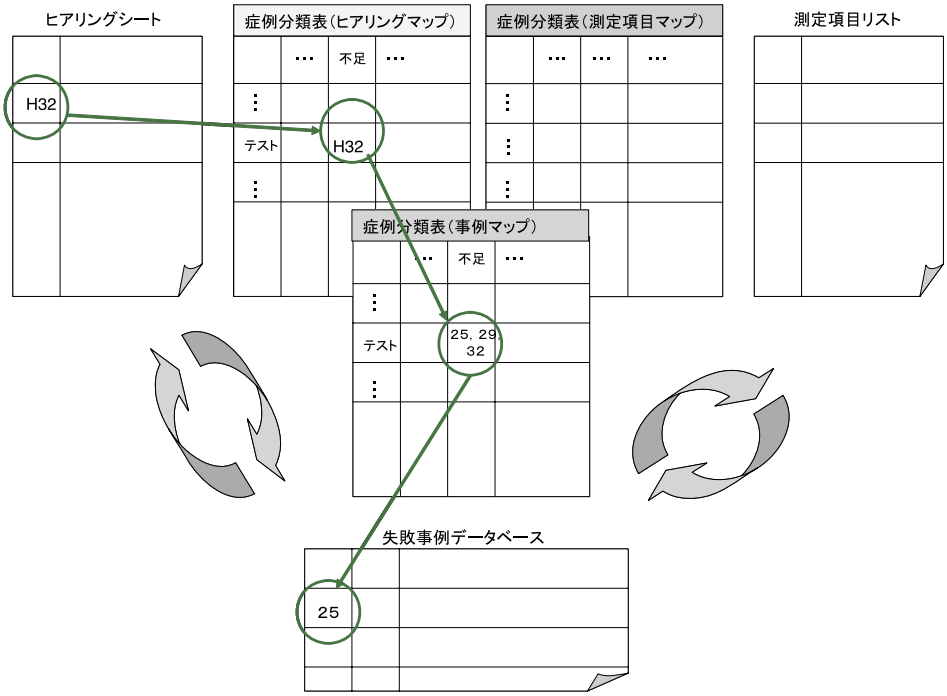
例えば、ヒアリングシートで「事前にテスト検証作業の手順・方法を計画した上で、テスト検証作業を実施しているか？」（チェック項目番号H32）という質問項目で問題の兆候が認められたら、次の手順で失敗事例データベースを参照して対策案を探す（**図表4-10**）。

⑴ ヒアリングシートのチェック項目番号H32が、症例分類表（ヒアリングマップ）のどこのセルにあるかを探す。

⑵ 症例分類表（ヒアリングマップ）にて該当の「H32」をすべて見つけ出したら、そのセルが属する行の左端（「テスト」）に注目する。これによって問題のカテゴリ、即ち問題の対象（「何が（何を）」）を知ることができる。

⑶ 次に、そのセルが属する列の最上段（「不足、不備（不良）」）に注目する。これによって問題の状態、即ち問題の症状（「どうした」に相当）を知ることができる。

図表4-10●症例分類表の活用方法③

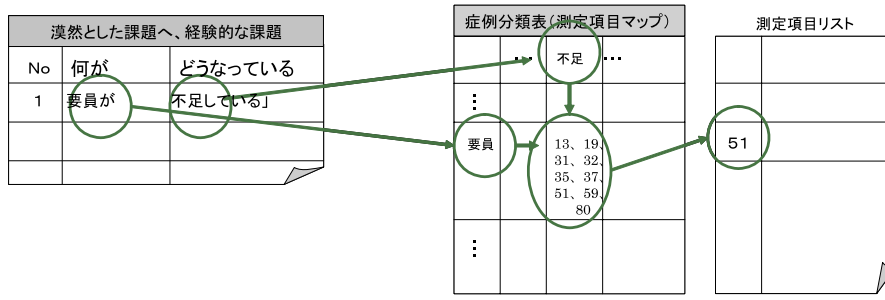


(4) 上記の問題の対象と問題の症状を組み合わせることにより、症例パターンを識別できる（「テストが不足」、「テストが不備(不在)」）。

(5) 症例分類表（事例マップ）を参照する（この表の詳細は【付録6 症例分類表】を参照）。症例分類表（ヒアリングマップ）の「テストが不足」と同じセルに記載されている事例番号を調べ（25、29、32）、失敗事例データベースで該当事例に目を通し、類似した事例（事例番号25）を選ぶ。

(6) 最後に、類似した事例（事例番号25）の対症療法（「テストの再実施」、「ソースコード・レビューの実施」）や再発防止策（「機能追加ルールの明確化」、「機能追加のリスクを顧客と合意」）を調べ、対策の候補とする。なお、ヒアリングシートで特定した問題に一致する事例が、必ずしも存在するとは限らな

図表4-11●症例分類表の活用方法④



い点に注意してもらいたい。

4.4.4 活用方法④「漠然とした課題などを症例分類表で捉え、測定項目を調べる」

プロジェクトにおいて、症例パターンがヒアリングや測定を行わずとも直感的に識別できることがある。例えば、「コミュニケーションが不足」しているケースや「キーパーソンが不在」であるケースなどは、なんとなくそのような症例を感じ取るものである。

また、企業の特性、組織の特性により、どのプロジェクトにおいても必ず同じことが問題となることがある。例えば、従業員が少ないソフトウェア開発会社では、どのプロジェクトでも「要員が不足」という症例が発生することは、経験的に感じ取っている。

このように症例の当たりをつけている場合では、症例をより具体的に分析し、不確かさを取り払うために症例分類表を使う。症例分類表（測定項目マップ）を介して測定項目を調べ、測定するのだ。開発フェーズの複数の時点で、定量的なデータを収集・分析していくことが問題を明らかにする上で有効である。

経験的に「要員が不足」することが潜在的問題の可能性として認識されていたら、次の手順で症例パターンを識別する（**図表4-11**）。

(1) 症例分類表（測定項目マップ）で、対象の「要員」の行と、症状の「不足」列を識別する。

(2) これによって症例分類表（測定項目マップ）の症例パターン（「何が」「どうした」が分かる）のセルを特定する。

(3) このセルに記載された測定項目番号（13、19、31、32、35、37、51、59、80）から、問題を発見するために役立つ測定項目の候補が分かる。

(4) 最後に、測定項目リストの「測定データの見方、分かること」を参考にして、より適した測定項目を選んで測定する。例えば「テスト経験者の数」（測定項目番号51）を選択し、「体制表やキャリアシート」から定期的に経験者数を測定する。

4.4.5 経験者による判断

症例分類表の活用方法①～④のような手順で症例パターンを絞り込めたら、さらに経験者の知見を用いた総合的な分析を行う。

例えば、測定項目リストの「開発チームごとの検出バグ（現象）数」（測定項目番号26）のデータを収集し、測定したところ、各開発チームでバラツキが見つかったとしよう。測定項目リストの「測定データの見方、分かること」欄には、①バグ発生数の多い開発チームには、品質基準が明確に伝わっていない可能性がある、②開発チームごとの品質への取り組み姿勢が分かる、と記載されている。①に関しては、ヒアリングシート「H57（協力会社の進捗・品質に関する中間フォロー実施計画が明確になっているか？）」を調査することが示唆されている。次に②については、症例分類表（測定項目マップ）から「要員の不備（不良）」、「協力・関係会社の不備（不良）」、「プログラムの不備（不良）」という症例が分かる。

このことから、開発チームごとに検出バグ（現象）数にバラツキが生じたのは、
「品質に関する中間フォロー実施計画が協力会社に明確に伝わっていたか」
「品質に関する要員の不備（不良）か」
「品質に関する協力・関係会社の不備（不良）か」
「プログラムの不備（不良）か」
という4点のいずれかに起因することが容易に分かる。

次に、ヒアリングを通じて、「計画そのものがあるか、誤りがないか」、「計画は正しいが、その通りに実施されているか」、「実施状況は管理されているか」などの視点でエビデンスに基づいて調べる。また、バグ検出に関わることから、テスト計画、テストケース、テストデータのレビュー記録を入手し、レビュー対象や参加者も調べることで、症例を絞り込むことができる。このような追加調査を通じ、事実に基づいて問題を特定する。

測定項目リストで測定した値から問

題の兆候に気付いたら、その要因を見極めるために、ヒアリングシートの複数の項目でプロジェクトの状況を調べ、正しく行われているものを除外し、症例を絞り込んでいく必要がある。

4.5 問題のレベルとその判定の考え方

ITプロジェクトの問題を予見できた場合、その問題に対する改善策を検討する必要がある。問題に対する改善策は、その問題の大きさやその時々の状況に応じて異なったやり方がある。その問題の大きさや状況の程度を合わせて「問題のレベル」と呼び、その考え方について説明する。

問題のレベルは、**図表4-12**のように改善策が影響する範囲によって、大きく3つに区分できる。カッコ内には医療との対応を示す。「①プロジェクト内対応が可能なレベル」は、そのプロジェクト内での対応で改善可能であること、すなわち「自己改善」できることを指す。これはプロジェクト・マネージャやプロジェクト・メンバーが気付きを得て改善できるレベルである。

「②社内での調整が必要なレベル」は、そのプロジェクト内の力では改善できないが、プロジェクトを実施している会社レベルで対応が可能なことを指す。例えば、プロジェクト・メンバーの増員や、プロジェクト・マネージャの交代といったレベルである。「③顧客との調整が必要なレベル」は、顧客を巻き込んで、システムの機能の見直しや期間の延長など、顧客の要求を変更するような対応レベルを指す。当然、これらの問題のレベルは、プロジェクトの置かれた状況や問題発見の時期にも依存する。

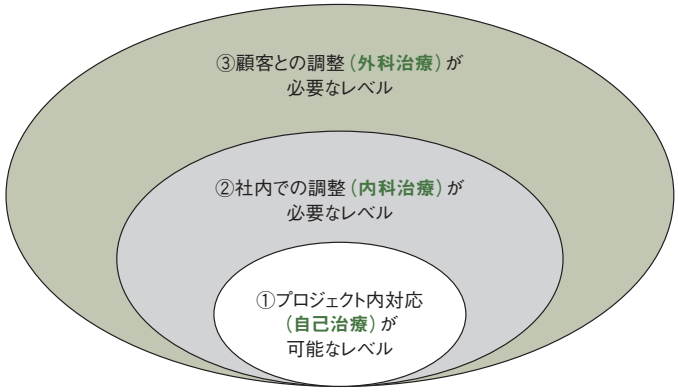
「問題のレベル」の判定は、プロジェクトの特性や判定時期などの複数の要素が複雑に影響し合って決まる。例えば、最も重要な要素であるプロジェクトの特性について、スケジュールが1カ月遅れた場合のことを考えてみよう。仮に、法律の施行時期とシステムの稼働時期が密接に関係しているプロジェクトのときは、法律の施行時期に間に合わない程の遅れであれば大きな問題になる。一方、社内利用に限られる情報系のシステム開発のとき、スケジュールの遅れはそれほど致命的な問題にはならない。スケジュール以外のプロジェクトの特性としては、「戦略的な重要性」、「市場的な重要性」、「技術的な新規性」、「プロジェクト・リスクおよびビジネス・リスクの大きさ」などがある。

問題の判定時期について言うと、例えばキーパーソンが病気で離脱した場合を考えてみよう。仮にキーパーソンの

離脱がプロジェクトの初期に起こったのなら、悪影響は交代要員を確保するまでの時期に限られる。交代要員を確保できれば大きな問題にはならない。

だが、総合テストの最中にキーパーソンの離脱が起こったら、納期に遅れが出るなどの大きな問題になりかねない。改善策の立て方は、こうした様々な要素を勘案して総合的に判断する必要がある。

図表4-12●問題のレベルとレベル分けの基準



問題のレベル	レベル分けの基準
①プロジェクト内対応（自己治療）が可能なレベル	追加要件が発生したが軽微な影響しかなくスケジュールを調整しなくても吸収できる場合のように、プロジェクト・マネージャの裁量範囲内で対応することができるレベル
②社内での調整（内科治療）が必要なレベル	想定以上の大量のバグが発生し、緊急にキーパーソンや大量の要員の増強が必要な場合のように、プロジェクト・マネージャの裁量範囲を超えた対応策が必要なレベル。しかし、社内調整の範囲内なので、外部のステークホルダーは対応策の実施に気が付かない
③顧客との調整（外科治療）が必要なレベル	致命的な機能不足リスクの発生によって納期を遅らせざるを得ない場合のように、顧客との調整が最優先となるレベル。プロジェクトのこの状態をクライシス（ITプロジェクトの最優先目標などを脅かすもの）と呼ぶ。リスクが発生したものすべてがクライシスになるわけではないが、コスト増、納期遅延、機能不足、品質不良、性能不足などが解消されず、予定稼働時期になってもシステムが正常に動作しないときには、解決行動を顧客と調整することが最も重要になる

第5章

直せる化

5.1 概要

発生したプロジェクトの問題に対して改善を行う必要があるが、実際にどのような対策をとるかはプロジェクトの状況に応じて千差万別である。それゆえ、これまでプロジェクトの問題は、プロジェクト・マネージャの経験に依存した改善策がとられがちだった。

プロジェクトの改善で特に課題となるのは、問題の兆候をつかんでおきながら、問題の大きさを見誤り、間違った対処をしてしまうことだ。後でかえって大きな問題となることがある。病気に例えると、風邪と勝手に判断し、適切な処置を施さなかったために、肺炎となってしまうようなケースである。

問題の大きさを見誤らないためには、適切に問題を特定することが最初の一歩であることは言うまでもない。これまで説明してきたプロジェクトの「見える化」を行って、「言える化」するというプロセスを経て、本章で説明する「直せる化」を行うことが重要である。

本章では、より適切な改善策をとっていくために、改善活動についてどのような考え方や視点があるか、そして改善策の適用で何に気を付けるべきかを説明する。これまで改善活動については各社各人の経験に裏付けられた知見に依るものとして、あまり議論されてこなかった。しかし、本章でいくつかの視点を示すことで、今後の改善活動のあり方についての議論の一助としたい。

5.2 改善活動の考え方と3つの視点

問題が発生した場合の改善策を議論する場合には、次のような考え方がある。現在発生している問題を正常な状態へ戻す「対症療法」と、今後、同様の問題が発生しないように予防する「再発防止」である。

再発防止は、「そもそもどうあるべきだったか」を議論し、次のプロジェクトにノウハウを活用していくことに主眼を置

いている。

この分野では、国際的に普及しつつあるCMMI（Capability Maturity Model Integration）やPMBOK（Project Management Body of Knowledge）などのベストプラクティスがまとめられている。SECにおいても、開発プロセス共有化部会などで標準プロセスが検討されている。また、本書がプロジェクトのリアクティブ活動に主眼を置いていることもあり、再発防止に関する考察は他の書籍や文献に譲ることとする。

一方の対症療法は、発生した問題についての対策を見つけ、より良い状態にもっていくことに主眼を置いている。しかし、現場においては、その場その場での“場当たり的な対処”や“属人的な対処”が行われており、かえって問題を複雑にしているケースも見受けられる。問題を見える化し、問題のレベルを的確にとらえ、そして適切な対処を行っていく必要がある。ここでは、この対症療法について考察する。

プロジェクトの改善方法として、実施可能な施策のパターンはそれほど多くはない。しかし、プロジェクトの問題への対応策としては、プロジェクトの性質や原因の複雑さなどによって、さまざまなバリエーションが考えられ、対症療法を完全にマニュアル化することは困難である。ここでは、改善活動の考え方を以下の3つの視点で整理する（**図表5-1**）。

視点① 改善活動に向けた「場」の形成
視点② 見える化手法を利用した改善
視点③ 改善策のパターン

5.3 視点① 改善活動に向けた「場」の形成方法

システム開発がプロジェクトとして人が行うものである限り、プロジェクトを「見える化」した後の改善活動もプロジェクト・メンバー自らが実施するというモチベーションを維持することが必要である。

医療に例えると、例えば体調に異変を感じた際に医者に診てもらい、医者の指導に基づき治療を受けたり治療薬を服用したりするなどの治療活動を本人が主体的に行う。プロジェクト活動でも同様に、プロジェクト・メンバーが自らの意志で問題を発見・診断し、改善していくという姿勢が必要である。

また、改善活動は、そのプロジェクト内のメンバーだけの活動で閉じるものではない。プロジェクトにかかわるステークホルダー全員が一丸となって取り組むべき活動である。

図表5-1●プロジェクトの見える化手法における「改善手法」の位置づけ

3つの視点とその概要	
改善活動に向けた「場」の形成	関係者を改善活動にいかにして巻き込むか、組織的な体質改善なども含めて、改善活動の雰囲気や環境を作っていく
見える化手法を利用した改善	チェックシートや測定項目、症例分類表群など、見える化手法の考え方やツールを基に改善ポイントを洗い出す
改善策のパターン	改善策として、どのような対策を取り得るのか、これまでの経験を基に、改善方法のパターンを形式知化する

しかし、プロジェクトの支援機能であるPMOが改善活動を行う際に、プロジェクト側からは「プロジェクトの監査」と受け取られ、プロジェクト・メンバーに警戒心を起こさせるようなことが往々にしてある。プロジェクトを救うための活動であること、そのためにはいくつかの「見えていないものを、見えるようにする」必要があることを、プロジェクトのステークホルダー間で共通認識として持つべきである。

このように、お互いに共通認識を持つために、一緒に改善していこうという「場（雰囲気）」を形成していくことは重要なことである。**図表5-2**に改善活動の場の形成例をいくつか示す。基本的なことではあるが、これらをきちんと実践できるかどうかで「場」の良し悪しはまったく変わってしまう。

このように改善活動の場の形成を通じて、プロジェクトのステークホルダーが一丸となって改善を実施していくという基盤や文化を築くことが最も重要である。たとえ方法論があったとしても、当事者がその気にならなければ、改善の効果は見込めない。

5.4 視点② 見える化手法を利用した改善方法

第3章と第4章で「見える化」手法における考え方やツールを説明してきたが、これらの考え方やツールなどを適用することで、問題がどこにあるのかを知り、改善策としてどのようなことが実施可能なのか気付くことができる。ここでは、手法・ツールをもとにした改善の「気付き」のポイントについて説明する。

図表5-2●改善活動の「場」の形成例

場・雰囲気	概要
上層部とプロジェクトの問題の共有化、遅滞ない課題解決への取り組みの場を作る	プロジェクト・マネージャだけで解決できる問題には限界がある。自分だけで対応するのではなく、上層部を巻き込むマネジメントを心掛ける。問題の芽に気付いたときは、上層部にアラームを上げて、先手先手の対応策を用意する。上層部も、問題の発生を突然聞くのではなく、事前の心づもりがあれば、パニックに陥ることなく、対処できる
プロジェクト・メンバーとプロジェクトの問題の共有化ができる場を作る	最初に問題に気付くのは担当者であり、それがプロジェクト・マネージャにすぐに伝えられるかどうかは、報告しやすい場になっているかに左右される。会議の場を設けるだけではなく、例えば開発者への個別面談を実施したり、問題をエスカレーション（報告）しやすい雰囲気を作る。問題が隠れたりくすぶったりする雰囲気を変えていく仕組み作りやメンタリングを行う必要もある
モチベーションが上がりやすい雰囲気を作る	先の見えない単調な作業（例えば単純なテスト作業など）は、作業者の精神的な負担が大きく、モチベーションが下がったり、作業品質が下がったりしやすい。そこで、例えば期限や作業範囲を明確に示したり、さらにその作業の意味・目的などを明確にしたりすることでモチベーションを上げられる。このように、モチベーションを上げていくことで、改善活動そのものを、メンバー自らが実施していく素養ができてくる
話をよく聞く	担当者は、常に自分の悩みを聞いてほしいとの願望を持っている。この願望にこたえるために作業報告の場を作り、じっくり話を聞くことが大切である。担当者の異変・兆候を早期に察知して対策を打つことがプロジェクトで生じる問題の初期消火になることが多い
組織の特質に応じた手順を怠らない	企業の中や企業間で何かの行動を行うときには、その組織の風土ともいうべき暗黙の手順が存在することがある。それはとても非合理的な手順かもしれない。しかし、その手順を行わないと、上司や顧客が抵抗勢力に変わったり、役員が承認できなかったりするなど思いのほかうまく行かないことがある。古い体質の企業、官僚的な体質の企業、一部のオーナー企業など、その組織のなりたちで違ったものになるので、いつでも同じ方法で対応できるわけではない
ステークホルダーを巻き込む会議体を運営する	最近のシステム開発では、要件の追加・変更の発生は当たり前であり、それをうまくマネジメントすることが重要になっている。問題が発生してから、顧客と協議するのではなく、要件管理、構成管理、進捗状況などの問題を共有するための会議体をあらかじめ設置する必要がある
主要メンバーのそれぞれがオーナーシップを持つ	プロジェクト・マネージャ、プロジェクト・リーダー、品質管理などの主要メンバーが自分の担当分野の問題を自分のものと捉えることがプロジェクトの成功の鍵であり、かつ自分の能力を高める動機付けとなる
プロジェクト・メンバーのすべてに、プロジェクトの進捗と問題を見える化する	プロジェクト管理ツールがあり、それを使えばプロジェクトの進捗状況を把握できるようにすることと、進捗状況が壁に張られ、見に行く努力をしなくても眼に入るようにするととはその効果が異なる。各メンバーが果たすべき役割、それへのコミットメント、そして達成状況を共有できる状況は、健全なソフトウェア文化を定着させる第一歩である

5.4.1 俯瞰図を用いた改善

俯瞰図による見える化を通じて、プロジェクト・マネージャは次のようにプロジェクト・マネジメントを改善することができる。

(1) このプロジェクトにおけるマネジメントのドミナント・アイテムが何か明確になる。
(2) 重点マネジメント・ポイントがどこかが明確になる。

5.4.2 自己診察を用いた改善

毎月決めた日に、すべてのプロジェクトに自己診察を実施させるとよい。この方法を通じて、プロジェクト・マネージャは次のようにプロジェクト・マネジメントを改善することができる。

(1) 経験の浅いプロジェクト・マネージャに、15の知識エリア（【付録7 知識エリア】を参照）ごとに、マネジメントのポイントがどこかを気付かせることができる。
(2) 軽微な問題であれば、自己評価シートの対応策を参考に自己診療が行える。

さらに、PMOは次のようにプロジェクト・マネジメントの改善を支援することができる。

(3) プロジェクトで各知識エリアに関するマネジメントの実施状況が分かり、支援すべき対象プロジェクトとそこで改善すべき課題が明確になる。
(4) 組織全体として見た場合に、マネジメントの弱い領域が分かる。マネジメントの弱い領域は、組織的に対応すべき課題であることが多い。
(5) 最近頻発するリスクについてのチェック項目を付加することによって、すべてのプロジェクト・マネージャにそのリスクへの気付きのヒントを与えることができる。
(6) 自己診察結果を時系列で把握することによって、組織のマネジメント・レベルの動向を判断でき、必要に応じて対応策を打てる。

5.4.3 専門家ヒアリングを用いた改善

専門家ヒアリングは時間と費用がかかるので、すべてのプロジェクトで実施することはできないが、実施すればプロジェクト・マネージャは次のようにプロジェクト・マネジメントを改善できる。

(1) ヒアリングやマネジメント資料の確認作業を通じて、マネジメントのポイントが明確になり、マネジメントのレベルアップができる。
(2) 自己評価と専門家ヒアリングの評価が乖離している場合、自己評価

の甘い個所に気付く。

5.4.4 定量データを用いた改善

定量データを使った見える化により、プロジェクト・マネージャは次のようにプロジェクト・マネジメントを改善できる。

(1) 問題への対策を打つ際に、どの時点で発生した問題かを明確にできる。
(2) 定量的に測定できるデータの変化の方向によって、問題になろうとしている状況を早期に把握して、対応策を実施できる。

5.4.5 失敗事例データベースを用いた改善

失敗事例データベースを使った見える化を通じて、プロジェクト・マネージャやPMOは次のようにプロジェクト・マネジメントを改善することができる。

(1) 失敗事例データベースを見ることによって、類似プロジェクトの問題の原因、結果と対応策が明確になる。
(2) 失敗事例データベースの事例と比較することにより、自組織の弱点が明確になる。

5.4.6 統合的アプローチ手法を用いた改善

統合的アプローチによる言える化を通じて、どの対象（例えばキーパーソン）のどのような症状（例えば、依存）が頻発しているかが分かる。この手法を用いることで、複数の原因が絡んだ問題を分析し、改善することができる。

5.5 視点③ 改善策のパターン

実際のプロジェクトの問題解決方法は多種多様である。例えば、納期が差し迫っているのに開発が計画通り進んでいないというケースでは、「では納期を延ばしましょう」という単純な対策に落ち着くことはまずない。

現実には、「納期に提供する予定だった機能のうち、主要な機能に絞って納期に提供する。その他の機能は順次リリースすることとして別の納期を設定し、サービスインをする。提供を遅らせた機能については、提供できるまでの間、人手などによる運用での回避策を考える」といった感じの対応を迫られるはずだ。

このように、プロジェクトの状況や構築しようとしているシステムの位置付け、顧客側の業務や運用能力に応じた対応策がとられる。

しかし、どのような対策も、いくつかの対策の組み合わせと考えることができる。どのような場合に、どのような対策が可能なのか、その対策の要素を洗い出しておくことで、実施できる改善

策を検討しやすくなる。

対策の要素を、症例分類表の対象の分類である「人」、「物」、「金」、「スキル」、「環境」、「その他」の観点で洗い出して抽象化すると、いくつかのパターンが見えてくる。以降、この一覧について説明する。各項目の対策の要素は、程度の軽いものから重いものへという順序で記述している。この表から分かる通り、取り得る対策は、分解してみるとそれほど多くはない。

まず、「人」に関する改善策のパターンを見てみよう（**図表5-3**）。プロジェクト・マネージャに関する問題の場合は、問題の程度が軽ければプロジェクト・マネージャに助言して「待つ」、その後の状況を「確かめる」という対策になる。プロジェクト・マネージャ自身の自助努力による改善を促す。

問題の程度が少し重くなってくると、マネジメントを支援するスタッフを補充したり、PMOが支援したりすることになる。さらに重くなるとプロジェクト・マネージャの交代を行うなどの手段がとられる。「人」に関する対策としては、人海戦術として社内外の要員を大量投入することもある。

「物」の改善策のパターンでは、例えば計画では完成済みであるべきテストケース表がない場合、それを「作る」ことから始める（**図表5-4**）。すでに完成しているが不備・不足がある場合、あるいはレビューができていないのであれば、再度レビューして「直す」、テストケースを「増やす」、記載内容を「増やす」、「強める」必要がある。また、不備が多い場合は、それを直すよりも、類似プロジェクトで使用したテストケース表に修正を加えたものに「替える」方が、早く必要なテストケース表を用意できる。このとき、すでに作成したテストケース表は「捨てる」ことになる。

「金」の場合では、予算が作成され、それに対する実績の超過が発生している場合、その事実を「確かめる」こと

図表5-3●改善策のパターン＜人＞

＜人＞		
●待機、養生、回復	→	待つ
●管理、監視、監査	→	確かめる
●教育、再教育、気づき	→	教える
●支援、補佐、補充	→	支える
●交代、代替、再編	→	替える
●縮小、削減、中止	→	減らす、止める
●増強、拡大、強化	→	増やす、強める

図表5-4●改善策のパターン＜物＞

＜物＞		
●作成、構築	→	作る
●修理、修復	→	直す
●改変、改造、改定	→	替える
●棄却、廃棄、縮小	→	捨てる、減らす
●増強、拡大、強化	→	増やす、強める

から開始する（**図表5-5**）。問題になっている出費があれば、その内容を調査して「詳しくする」。また、人員の追加や物品の購入など、必要なコストに対しては、そこにかける金額を「増やす」必要が生じることもある。予算の総額を変更せず、あるリソースにかけるコストを増やした場合は、内訳を見直し、用途を「替える」、他の項目の金額を「減らす」といった改善をする。究極の減額方法として、その計画自体を「捨てる」必要がある場合もある。プロジェクトのスコープ変更が発生した場合は、顧客との間で計画変更を実施し、予算の総額を増やすために金を「作る」という対策もあり得る。

「スキル」の面では、プロジェクト・メンバーの技術あるいはテスト・スキルが不足している場合に「教える」、「支える」など本来のプロジェクト推進方法が考えられる（**図表5-6**）。だが、サービスインまでの時間がなく、「教える」時間がない場合には、緊急対策として要員を「増やす」、「替える」ことも検討しなければならない。

「環境」の改善パターンは、テスト工程に入り、テスト環境がなければ「作る」ところからスタートする（**図表5-7**）。計画に従って、ハードウェア、ネットワーク、要員などを手配する。また、テスト工程の進捗の遅れがテスト環境の不足に起因するなら、残テスト項目数、残工期、要員数に応じてテスト環境を「増やす」ことになる。

人、物、金、スキル、環境以外の面でも、**図表5-8**のような改善策のパタ

図表5-5●改善策のパターン＜金＞

＜金＞		
●作成、実施	→	作る
●確認	→	確かめる
●変更	→	替える
●詳細、具体	→	詳しくする
●縮小	→	減らす
●破棄	→	捨てる
●増強、拡大、強化	→	増やす、強める

図表5-6●改善策のパターン＜スキル＞

＜スキル＞		
●管理、監視、監査	→	確かめる
●教育、再教育	→	教える
●支援、補佐、補充	→	支える
●交代、代替、再編	→	替える
●増強、拡大、強化	→	増やす、強める

図表5-7●改善策のパターン＜環境＞

＜環境＞		
●構築	→	作る
●改築、改造、改定	→	替える
●増強、強化	→	増やす、強める
●廃止、縮小	→	止める、減らす

ーンがある。これら改善策のパターンを表形式でまとめたものが**図表5-9**だ。この表を参考にすることによって、プロジェクトの問題が発生した場合に、どのような視点で改善策を考えるべきかを確認できる。この表には具体的な改善策が記述されているわけではないが、改善策の抜けやもれを防げると考えている。

図表5-8●改善策のパターン＜その他＞

＜その他＞		
●待機、養生、回復	→	待つ
●管理、監視、監査、確認	→	確かめる
●詳細、具体	→	詳しくする
●教育、再教育、気づき	→	教える
●支援、補佐、補充	→	支える
●交代、代替、再編	→	替える
●修理、修復	→	直す
●作成、構築、実施	→	作る
●縮小、削減、中止、破棄	→	減らす、止める、捨てる
●増強、拡大、強化	→	増やす、強める

図表5-9●改善策のパターン表

	＜人＞	＜物＞	＜金＞	＜スキル＞	＜その他＞
増やす、強める →増強、拡大、強化	○	○	○	○	○
待つ →待機、養生、回復	○				
教える →教育、再教育、気づき	○			○	
支える →支援、補佐、補充	○			○	
確かめる →管理、監視、監査、確認	○		○	○	
替える →交代、代替、再編	○		○		○
減らす、止める、捨てる →縮小、削減、中止、破棄	○	○	○		
作る →作成、構築、実施		○	○		○
直す →修理、修復		○			
替える →改変、改造、改定		○			
詳しくする →詳細、具体			○		

第6章

「見える化」に関する研究の解説

これまでの章では、ITプロジェクトの「見える化」を実践するための方法をできる限り具体的に示してきた。しかし、これらの実践方法が実際に効果を発揮するためには、見える化に必要とされる様々な基礎データをいかに低コストで正確に収集するかが肝要だ。さらに、そこから見える化に至る直前までの客観的な分析を、いかに機械的かつ的確に行うかが鍵となる。本章では、そういった見える化を実践するための“インフラ”に相当する部分を、強力にサポートする最新技術をいくつか紹介する。

まず、これまでの章でも何度か言及しているEASEプロジェクトの成果を紹介する。EASEプロジェクトは、2003年度から文部科学省のリーディング・プロジェクト「e-Society基盤ソフトウェアの総合開発」の一環としてスタートした、奈良先端科学技術大学院大学の鳥居宏次学長を代表とする産学連携プロジェクトである。

EASEプロジェクトは、信頼性や生産性に課題の多いソフトウェア開発において、科学的根拠に基づく開発手法であるエンピリカル・ソフトウェア工学（Empirical Software Engineering）の確立を目指している。その基本理念は、開発現場から得られる「現実に即したデータ」を基に「科学的な分析や検証」を行うことである。現実と離れた理論的なモデルのみを扱う研究ではなく、かといって理論や実験による裏付けのない技術論でもなく、従来のソフトウェア工学が抱えてきた理想と現実（つまりは学と産）の間のギャップを埋めるための取り組みとなっている。

本章では、EASEプロジェクトの成果である開発データ収集ツール「EPM（Empirical Project Monitor）」や、SECのCOSEプロジェクトにおけるEPMの適用事例、さらにEPMなどから得られるデータを分析する応用技術として、コードクローン分析や協調フィルタリングを解説する。

また、これらに加えて、SECの見える化部会での最新の取り組みとして、失敗事例の分析とその傾向についても紹介し

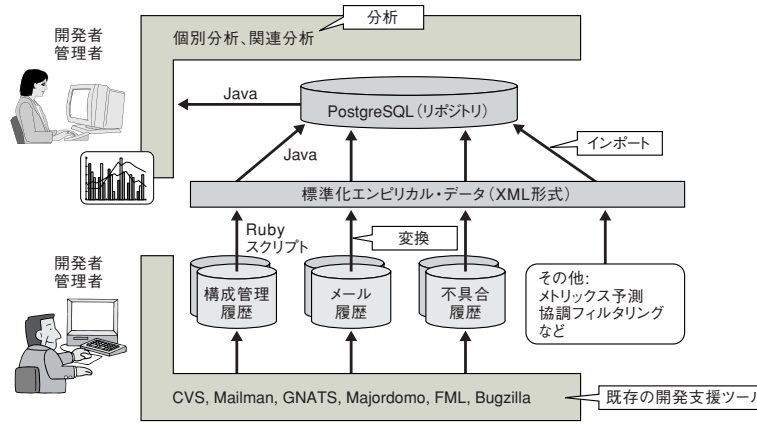
たい。

6.1 EASEプロジェクトのEPMツールを使ったプロジェクト定量化

ソフトウェア開発におけるプロジェクトの定量化の重要性は誰しもが認めるところであろう。しかし、計測方法に問題があるならば、その役割を十分に果たすことはできない。EASEプロジェクトで開発したEPMは、ソフトウェアの開発中に利用される開発支援ツールの利用履歴を収集・分析することで、必要なときに随時確認可能な、改ざんの少ないデータを収集・分析することができる。

では、従来のデータ収集方法との違いを見てみよう。従来のソフトウェア開発でデータを収集する場合、その多くが人手によるものだった。このため、データ収集作業の負担を嫌う開発者は、新たなデータの収集にはなかなか協力してくれるものではなかった。EPMは、バージョン管理ツールCVS、メーリングリスト管理ツールMailman、Majordomo、FML、障害管理ツールGNATS、Bugzillaといった開発支援ツールの利用履歴からデータを直接収集するため、開発者にデータ収集の負担をかけずに済む（**図表6-1**）。これら以外の開発支援ツールを使っている場合でも、EPMのインタフェース（オブジェクト指向スクリプト言語Rubyで記

図表6-1●EPMツールの概要



述したプログラム）を変更あるいは追加することで、EPMを利用できる。

また、従来は工程の区切りを中心に、週次・月次といった一定期間ごとに開発データを収集し、次の工程に進んでよいかどうかの判定や、現在の状況の確認を行っていた。判定や状況確認などの作業に合わせた合理的なデータ収集方法ではあるが、問題の発生をリアルタイムに発見することはできない。EPMでは、必要と思われるときにデータを収集・分析できるほか、サーバーの設定によって夜間などにデータを収集しておき、任意の時点で分析を行うことが可能である。

次工程に進む際に手作業で収集したデータを用いることは、収集したデータの信憑性に影響を与えているかもしれない。次工程に少しでも早く進みたい開発者は、データ収集の目的を知っている。悪意はないとしても、開発者が都合のよいデータを作ってしまうかもしれない。EPMが利用するデータは、開発支援ツールの利用履歴を用いるので、開発者が自分の作業に集中しているならば、そのデータは比較的信頼できるものであると考えられる。

このようにEPMは、開発者に負担をかけず、必要な時期に、改ざんの少ないデータを収集する。そのデータを基に、時系列のグラフによる状況把握、詳細情報の一覧、パレート図による問題分析、信頼度成長曲線による残存バグ予測などを行う。

6.2 チェックシートのCOSEプロジェクトへの適用

6.2.1 適用対象プロジェクト

見える化施策の試行対象として、政府予算による中規模プロジェクトへの適用が決まった。それは「プローブ情報プラットフォーム」と呼ぶ実験用システムを開発するプロジェクトで、2005年春にスタートした。このシステムは、乗用車やバス、トラック、タクシーなどにセンサーを搭載し、その車両（プローブカーと呼ぶ）の位置情報などを収集して高精度な交通関連情報や運転支援情報をリアルタイムで提供することを目指している。システムの開発とそれを使った評価実験は、「ソフトウェアエンジニアリング技術研究組合」が進めている。この研究組合は、自動車メーカー、関連機器メーカー2社と大手ソフトウェア企業5社が法律に基づいて組織した。ソフトウェアエンジニアリング技術研究組合を英文で表記すると「COnsortium for Software Engineering（COSE）」であるため、このシステム開発と実験のプロジェク

トは「COSEプロジェクト」と呼ばれている。

開発期間は2年間で、この間に2回のシステム開発と実験が行われる。ソフトウェアの開発は、研究組合の大手ソフトウェア企業が分担し、広域マルチベンダー開発となる。

研究組合を構成する各企業は、ターゲット・システムの分野においてライバル関係でもある。この計画ではお互いに競争領域と協調領域を峻別し、競争領域では情報を秘匿し、協調領域では情報共有を図りながら進めている。開発対象は、Linuxサーバー上で動作するC/C++のデータベース・アプリケーション・ソフトウェアと、情報表示用に使うPC上のソフトウェアである。

6.2.2 チェックシート適用状況

SECはCOSEプロジェクトに対し、チェックシートによるデータ収集と分析を実施し、その結果をプロジェクトの途中でフィードバックすることを試みた。具体的には、プロジェクト・マネージャ、各社リーダーにヒアリングするためのヒアリングシート（80項目）、これに先立つ予備調査用の自己評価シート（30項目）を開発し、自己診察とヒアリングによりプロジェクトの情報を収集した。この調査から、このようなヒアリングでしか収集できない多くのプロジェクト情報を得ることができた。

調査は2005年8月下旬、各開発会社が単体テストを実施しているくらいの段階で、各社間の結合テストを開始する少し前の時期に実施した。調査結果をレーダーチャートと簡単なレポートに要約し、9月下旬にフィードバックと意見交換の場をもった。調査とフィードバックは、組合企業に対して個別に行った。

ヒアリングは、より深い接点を求めて、COSEプロジェクトの参加各社に出向いて行った。その所在地は東京都内数カ所のほか、神奈川県、茨城県、愛知県などに分散しており、日程調整も含めて大仕事であった。

ヒアリングへの参加者は企業によって異なった。各社は複数階層の協力会社と開発しており、ヒアリングには各階層の代表的なメンバーのほか、企業によってはサブリーダー・クラスが参加した。この参加メンバーの選定には議論の余地がある。本来は各企業を代表する一人にヒアリングすべきで、そこで欠けていることを順に階層を下げて明らかにすべきである、という意見がある。このほうが、組織の中での責任分担状況が浮き彫りになる利点がある。しかし、現実にはこのようなきめ細かいヒアリングの設定は難しい。今回のように、調査の分解能は落ちるが各社の

核になるリーダー、サブリーダーに集まってもらった場での質疑が現実的である。この方法でもマクロな課題を明らかにできた。

実際にヒアリングすると、机上で考えたチェックシートの課題が浮き彫りになり、回数を重ねながら改良していくことができた。あらかじめ自己診察を行い、エビデンスとなる資料を用意してもらったが、その状況は各社によって異なった。ヒアリングの場で資料を取り寄せてもらうこともあった。

6.2.3 考察

ヒアリング時間は2時間を目標としたが、実際には2時間半から3時間を要した。実感として2時間を超えると長いと感じられた。

ヒアリングは全般に良い雰囲気で進められ、忌憚のない意見、情報を得られた。これは今回の調査が、ソフトウェア・エンジニアリング研究の視点からのもので、査察的なイベントではないためと推定される。厳格な雰囲気で実施して公式的な情報を収集するか、柔軟に実施して本音の状況把握をするか、ヒアリング運営の課題を感じた。

フィードバックの会合も各社個別に実施した。各企業にとって、全般的にはすでに把握されていたことの追認の要素が大きかったようである。それでも各企業の感想として、このヒアリングでいわゆる「気付き」を得た項目もあったということである。

実施側には各社のプロジェクトへの取り組み姿勢、組織構成上の特徴と課題、プロジェクト運営への意見など、他の手段では得られない情報を多数収集できた。特に各企業の開発体制、責任体制、それに伴うプロジェクト運営の状況を把握できた収穫は大きい。これらは研究組合方式、あるいは大型のマルチベンダー開発ではほとんど秘匿されているブラックボックスの情報である。

今回はこれらの情報を、全体のプロジェクト・マネージャとの意見交換を介してプロジェクト運営に反映しただけでなく、チェックシートの改良に反映できた。

6.3 COSEプロジェクトへのツール適用例

6.3.1 COSEプロジェクトで採用した6つの測定技術

COSEプロジェクトでは、EASEプロジェクトと共同で多次元のデータ収集を試み、その分析結果をプロジェクト運営にフィードバックした。プロジェクトの測定では、次のような6つの測定技

術と方法を採用した。

(1) EPMによる測定と分析

測定プラットフォームEPMを用いて、開発プロセスとプロダクトの基本情報を取得した。そのために、COSEプロジェクトでは、各社の開発管理環境をCVSとGNATSに統一した。EPMにより、例えばプログラム行数の推移、累積障害件数、残留障害数の推移、障害平均残留時間の推移、開発会社間メール件数の推移、これらとチェックイン、チェックアウトのタイミングとの関係、チェックアウト数との関係などを、ビジュアルな形で取得できた。

(2) レビュー記録の収集と拡張EPMによる測定と分析

専用の電子フォームを用いて、基本設計と詳細設計に対するレビュー記録を収集した。また、EPMの分析機能を拡張して、レビュー記録とCVSリポジトリのファイル更新に対して様々な分析を試みた。

(3) CCFinderによるコードクローン分析

コードクローンは、ソースコード中の類似するコード片（あるコードをコピーして編集したものや意図的に同一処理を繰り返し書いたもの）のことである。このクローンの分布状況、含有率などからプロジェクトの特徴を推し量ることができる。COSEプロジェクトではソースコードの提供を受け、大阪大学と神谷年洋氏の、CCFinderと呼ぶツールを用いて、ソースコードからコードクローンの含有状況、含有率を分析した。ソースコードへのクローンの含有状況は、各種のクローン関係メトリックスと合わせて、クローン散布図というチャートによって鳥瞰した。なお、コードクローンの詳細は【6.4 コードクローン分析】を参照してもらいたい。

(4) ベンチマーク・データベースと協調フィルタリング・ツールの応用による分析

プロジェクト・マネジメントに関する400項目のベンチマーク・データを、専用の入力フォームを使って収集した。これらの項目は、国内の有力企業が収集している項目や、ISBSG（International Software Benchmarking Standard Group）が収集している項目を参考にしながらSECが定めたものである。今回のプロジェクトでは、基本設計終了時に全体のプロジェクト計画値とそこまでの実績値を集めた。プロジェクト終了時に、残りの実績値を収集した。

プロジェクト途中のベンチマーク・データから、プロジェクトの将来に関す

る予測を試みた。SECが収集した1000プロジェクトの過去データを基に、奈良先端大で開発した協調フィルタリング技術を応用したツールを用いて類似のプロジェクト・グループを探し出し、予測に利用した。協調フィルタリングは、欠損のあるデータセットから類似のデータをグルーピングする技術である。

(5) プロジェクト・マネージャ、リーダーへのヒアリング用チェックシートの開発とヒアリングによるコンテキスト情報の収集

先に紹介したように、プロジェクトの管理体制などコンテキスト情報の収集を目的に研究組合の各企業を訪問し、企業内のプロジェクト・マネージャや開発リーダーにSECの作成した約80項目のチェックシートによるヒアリング調査を実施した。このヒアリングに先立ち30項目のチェックシートによる自己診察も行い、ヒアリング結果と比較した。このヒアリング調査を通して、EPMのような自動収集ツールでは収集できないコンテキスト情報を得られた。

(6) プロジェクト会議への継続参加によるコンテキスト情報の収集と分析への反映

さらに各種のコンテキスト・データを得るために、測定グループのメンバーの一部が、プロジェクトを通してプロジェクト会議に参加。自動収集では得られない情報を取得した。

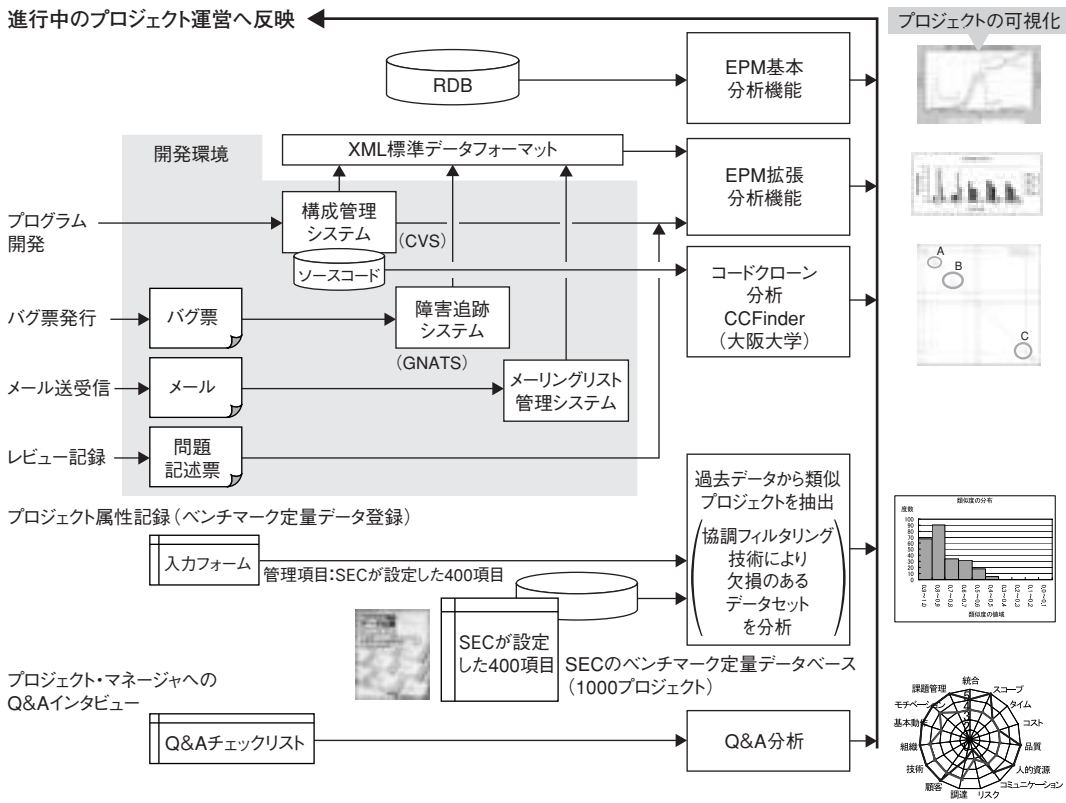
6.3.2 プロジェクト測定体制

COSEプロジェクトでの測定とフィードバックの体制を**図表6-2**に示す。収集データは結合テストやシステムの評価実験の場所とは物理的に離れたSECへ集められた。研究組合各社の情報秘匿に配慮したためである。

COSEの開発担当各社は、自社開発担当分の構成管理にCVSサーバーを使い、ソースコードを含むリポジトリをSECに定期的に送付した。開発中は各社内で障害管理ツールGNATSを運用し、そのデータを定期的にSECに送付した。開発会社間の結合テストでは全社共通のGNATSサーバーを用いて障害管理し、そのデータをSECに提供した。SECは収集データを厳重に管理し、EPMをはじめとする各種ツールで分析した。

分析結果は定期的にプロジェクト・マネージャと研究組合各社へ提供した。プロジェクト・マネージャに対しては全社の状況が俯瞰できる形で、各社に対しては自社の状況が分かる個別の分析結果の形で示した。ただし、プロジェクト・マネージャに対してもソースコー

図表6-2●COSEプロジェクトでの測定とフィードバックの枠組み



ドなど詳細な情報は秘匿した。

6.3.3 プロジェクト測定で明らかになったこと

一般に研究組合方式では、開発会社間の結合テストが始まるまで各社は開発状況を自己申告するだけで、その開発内容は秘匿される。今回、従来はブラックボックスだった各社の開発過程について、マネジメントに反映するべき基礎情報を取得し、具体的なマネジメントに反映することができた。

例えば各社の開発状況を、次のようにして定量的に把握することができた。まず、ソースコード行数の推移や障害件数の推移を追うことで、プロジェクトの進捗状況、開発量を俯瞰できた。また、ソースコード内のコードクローン

の含有率や分布から、ソースコードの素性や、開発グループの特性を推し量ることができた。各社が開発したソースコードから、まったく新規に開発されたコード、大規模な流用部分を持つコード、経験の浅い要員によるコードなどを識別でき、コード全体の保守性に関する懸念事項やリファクタリングの状況などを知ることができた。

ソースコードを秘匿した状態ではあるが、プロジェクト・マネージャと各開発会社がコードクローンの分析データを共有して議論することは非常に有用だった。コードクローン分析が、プロジェクト・マネージャがソースコードの特徴を把握するときに役立つだけでなく、開発会社にとっても利用価値があることが分かった。例えばある開発会社は、測定されたコードクローンに対して、「将来の機能の進化を考えて、あえてクローンにしてある」という設計思想を、自信をもって主張することができた。

コードクローンに関する別の例として、工程の後段で新たに大きなコードクローンが出現した場面があった。これは、類似の機能が2つある処理で起こった。初めから2つの機能仕様がきちんと確定していれば、当然共通化してコードを一本化したはずだったが、類似した機能のうち、片方の仕様が開発会社間の調整などで手間取り、仕様の細部が長らく保留のままとなっていた。この仕様が確定した時点で、先行開発した片方の機能はテストが終了していた。このため、開発スケジュールの都合でやむなく、先行開発した機能をコピーし、もう片方の機能を実装した。これがコードクローンとして測定されたのである。仕様の確定遅れによる影響が出たことを開発会社が主張できるだけでなく、当該機能の保守に関する注意事項を関係者で共有できた。

このようにコードクローン分析を介したコミュニケーションは、開発会社にとって設計書に示された機能を実装すること以上のやりがいを生み、高いモラルをもたらすことが期待できる。

このほかCOSEプロジェクトで実施した測定により、次のような点が明らかになった。

①CVSリポジトリのファイル更新状況を様々な角度から分析することで、開発状況を推測できた。ウォータフォール型で開発が順調に進んだこと、一部に試行錯誤型の開発があったこと、後工程での設計変更や障害判明の影響の大きさなどが、非常によく分かった。

②障害分析により、測定した各種の要因と障害の関係を分析できた。特に、障害が混入する工程の分析は、開発工程全体の評価に役立った。

③協調フィルタリング技術を応用したベンチマーク・データの分析では、工程の早い段階で過去データベースを検索することにより、ある種の予測や予告を行えた。

④レビュー記録の分析から、レビューに取り組む各社の姿勢の粗密が分かった。ウォータフォール型工程を組んでレビューを重視している企業と、レビューよりも後工程のテストを重視している企業に色分けできた。異なるレビュー文化の開発会社がモジュールを持ち寄って開発会社間の結合テストを実施する際、そこで起こり得る摩擦に備えることができた。

6.4 コードクローン分析

6.4.1 コードクローンとは

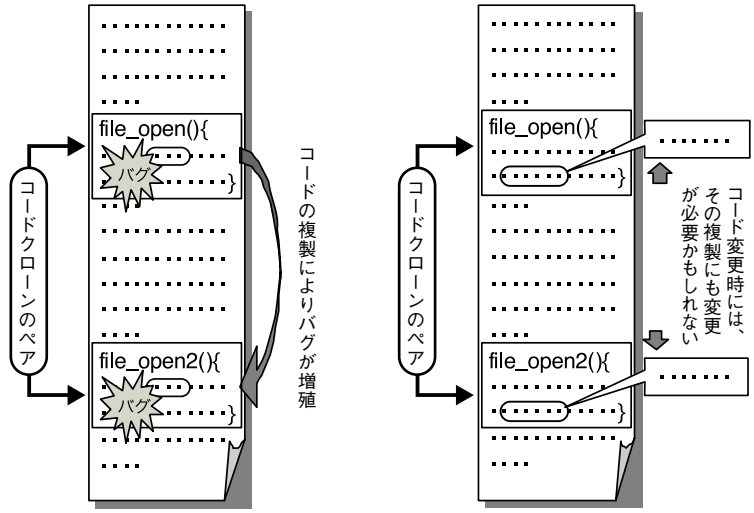
ソフトウェアの保守を困難にする要因の一つとして、近年、コードクローンが注目されている。コードクローンとは、「ソースコード中に散在する同一または類似した複製コード群」のことであり、コピー＆ペーストによるコードの流用など、様々な理由により生成される。

図表6-3は、Javaプログラム中のコードクローンの例である。3～9行目と11～17行目は、変数名やメソッド名を

図表6-3●コードクローンの例

```
 1. static void foo() throws RESyntaxException {
 2.   String a[] = new String [] { "123,400", "abc", "orange 100" };
 3.   org.apache.regexp.RE pat = new org.apache.regexp.RE("[0-9,]+");
 4.   int sum = 0;
 5.   for (int i = 0; i < a.length; ++i)
 6.     if (pat.match(a[i]))
 7.       sum += Sample.parseNumber(pat.getParen(0));
 8.   System.out.println("sum = " + sum);
 9. }
10. static void goo(String [] a) throws RESyntaxException {
11.   RE exp = new RE("[0-9,]+");
12.   int sum = 0;
13.   for (int i = 0; i < a.length; ++i)
14.     if (exp.match(a[i]))
15.       sum += parseNumber(exp.getParen(0));
16.   System.out.println("sum = " + sum);
17. }
```

図表6-4●安易なコードの複製は保守性の低下を招く



置き換えただけで、基本的な構文は全く同じである。このように、コードクローンは、単にテキストをコピーしただけではなく、変数名や関数名などを書き換えた形で存在する。

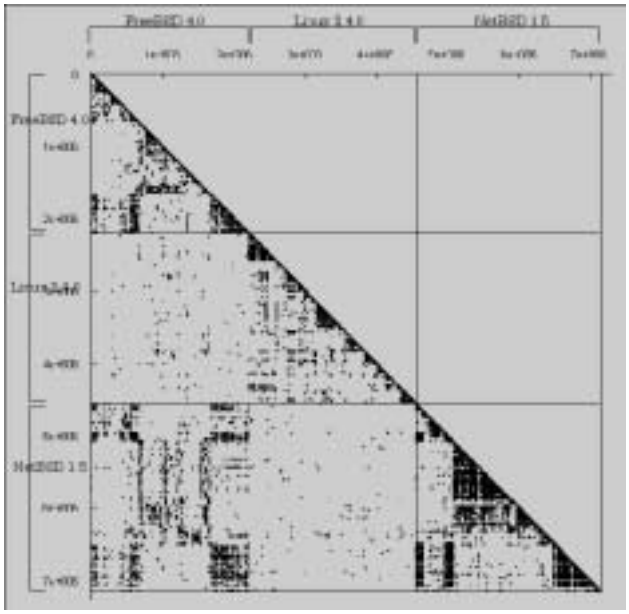
コードクローンの存在は、しばしば保守作業において障害になる（**図表6-4**）。誤りを含んだコードを複製することで、その誤りが散らばってしまうことや、ある部分に機能を追加する場合にも、そのクローンを探して同様の変更を施す必要があるか調べなければならないためだ。もっとも、すべてのクローンが悪いとは限らない。例えば、コード生成ツールが作成したコードや、パフォーマンスを改善するためにループの中身を展開したコードなどは、意図的に作成されたクローンである。

6.4.2 コードクローン検出・分析ツール、CCFinderとGemini

コードクローンを検出するには、単純なテキストの比較ではなく、構文レベルでの比較が必要である。このため、従来のコードクローン検出ツールで数百万行といった大規模なソフトウェアを解析するには長大な時間が必要とされていた。

大阪大学大学院情報科学研究科・ソフトウェア工学講座（井上研究室）で開発した「CCFinder」は、特許出願済みの高速アルゴリズムを用いること

図表6-5●クローン散布図の例
（FreeBSD、Linux、NetBSD のカーネルコードの比較結果）



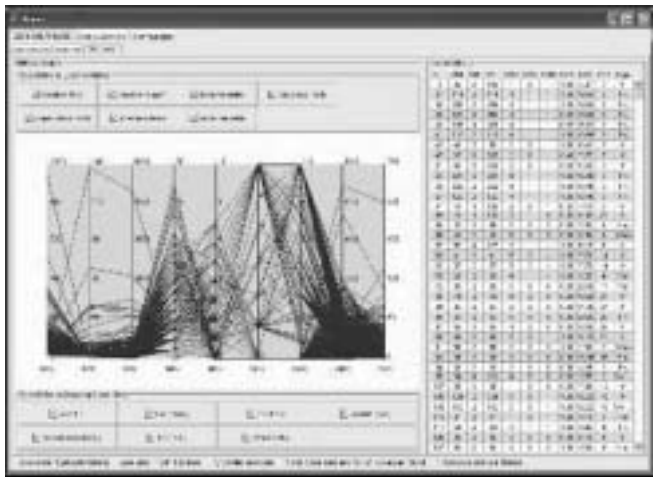
で、数百万行規模の大規模なソフトウェアに対しても実用的な時間内にクローンを探索できる。

CCFinderはコードを含むファイルセットに対して、構文レベルでの相互比較を実行して、検出されたクローンの位置をテキスト形式で出力する。より直観的にクローンの分布状況を確認するためのGUIフロントエンドとして、「Gemini」と呼ばれるツールも併せて提供されている。Geminiは、クローン散布図（スキャタープロット図）の表示機能や特定の性質を持ったクローンを絞り込んで表示する機能などを備える。

図表6-5は、Geminiが出力するクローン散布図の例である。縦軸と横軸はソースコードの累積行数を示し、クローンが検出された場所（行）に黒点がプロットされている。黒点の集中する部分ほどクローンの密度が高いことを示している。

図表6-6は、特定の性質をもつクローンのみを絞り込んで調査するための画面である。クローンの長さや出現数、散らばり具合などの属性（クローン・

図表6-6●メトリクスグラフによる分析画面例



メトリックス）の範囲を指定できる。多次元並行座標表現と呼ばれる形式のグラフを用いて、各メトリックスの上限・下限を指定し、絞り込まれたクローンのみが表示される。

6.4.3 コードクローンによる「見える化」の実践

コードクローン分析技術は、ソフトウェア保守における潜在的リスクの一部を「見える化」するうえで、非常に有効である。あるソフトウェア開発会社の持つCOBOLのレガシーコードに対して、CCFinderによるクローン検出結果と保守作業との関連を調査してみた。その結果、一定以上の割合でクローンが含まれているモジュールや、非常に長いクローンが含まれているモジュールでは、修正回数が他のモジュールより高いことが分かった。コードクローンが実際に保守性に影響を及ぼしていること示しているといえよう。

また、SECのCOSEプロジェクトでCCFinderを利用したところ、このツールはプロジェクト・マネージャがブラックボックスとなっているソースコードの特徴を把握するのに役立つとの評価を得ている。

CCFinderは、企業での試用・適用が進むとともに、学術分野でも国際的なデファクトの地位を獲得している。CCFinder、Geminiの詳細は、http://sel.ics.es.osaka-u.ac.jp/cdtools/index.htmlを参照してもら

いたい。また、多言語対応や処理速度向上を図ったCCFinderの次期版CCFinderXがIPAの未踏ソフトウェア創造事業の支援を得て開発されている。CCFinderXの詳細はhttp://www.ccfinder.net/ccfinderx-j.htmlを参照してもらいたい。

6.5 EASE協調フィルタリングによるプロジェクトの特性予測

6.5.1 過去プロジェクトのデータを用いた予測の問題点

プロジェクト特性の予測とは、ソフトウェアの開発工数やプロダクト品質（例えば出荷後のプロダクトに潜在するバグの数）など、プロジェクトの特性を表す数値（プロジェクト特性）を高い精度で予測することだ。これを利用すれば、人的リソースの割り当てや品質保証の効率化を実現できる。プロジェクト特性を予測するとき、従来は過去のプロジェクトのデータから重回帰分析などにより予測モデル（数式）を構築する方法が用いられてきた。例えば、工数予測モデルとして有名なCOCOMOもその一つである。**図表6-7**に、予測モデルの構築と利用手順の流れを示す。この例では、過去プロジェクトA～Cのデータを用いて予測モデル（ここでは重回帰式）を構築し、次に現行プロジェクトのデータをモデルに代入することで工数を予測している。

このような手法で精度の高い予測結果を得るには、組織内で統一されたポリシーを策定し、多くのプロジェクトから大量のデータを収集することが理想的である。しかし、これは容易なことではない。実際にデータを収集してみる

図表6-7●重回帰モデルを用いたプロジェクト特性の予測例

	開発言語	開発種別	概算FP	要員数	開発工数
過去プロジェクトA	1（COBOL）	0（保守）	100	40	250
過去プロジェクトB	0（Java）	1（新規）	35	8	40
過去プロジェクトC	0（Java）	1（新規）	25	6	35

ステップ1. 予測モデル（数式）を構築する

177 × 開発言語 ＋ 0.5 × 概算FP ＋ 23 ＝ 開発工数

ステップ2.
値を代入して予測する

現行プロジェクトX	0（Java）	1（新規）	30	10	予測値：38

と、欠損部分（データが未記録になっている穴あき部分）が多数あるのだ(**図表6-8**)。

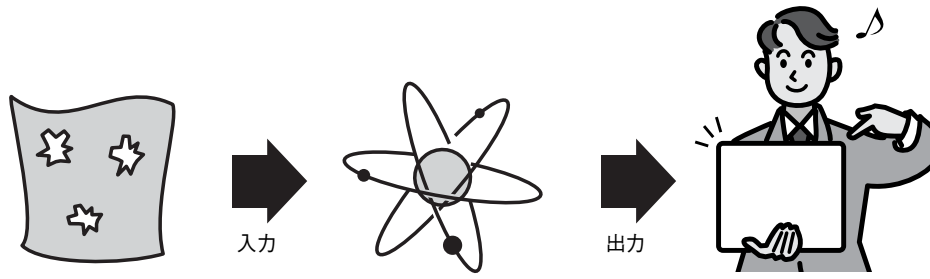
その原因を突き詰めると、開発現場での止むを得ない事情に行き着く。納期直前の切羽詰った時期に、プロジェクト・メンバーには不要なデータを収集する余裕があるだろうか。昼夜を問わず働いている部下に、「この部分のデータが収集できていないよ」と指摘できるだろうか。止むを得ない事情が積み重なることで、組織のデータ収集ポリシーが歪み、欠損部分ができてしまう。

図表6-8●苦労して集めたデータでも欠損（穴）が多数存在する

このように部分的に欠損したデータ（欠損データ）を用いた場合、従来方法では予測精度が大幅に低下してしまう。例えば、重回帰分析などの統計手法で予測モデルを構築するためには、欠損部分が存在しない完全なデータが必要になる。このため、モデルを構築するときは、欠損部分を含むデータを削除するか、欠損部分を何らかの数値で補完しなければならない。

しかし、こうしてデータに手を加えると、生データにはなかった人工的な“歪み”ができてしまう。結果として、構築された予測モデルの品質が劣化し、その精度も低下する。海外の研究者らの報告によると、欠損部分が全体の30%を超える場合、予測精度が著しく低下するというが、企業でデータを収集し

図表6-9●協調フィルタリングなら、欠損データを入力しても正確に予測可能



た場合、欠損部分が30％を超えることは珍しくない。

6.5.2 EASE協調フィルタリング法によるプロジェクト特性の予測

EASEプロジェクトでは、データ欠損に対するロバスト（頑健）な予測手法として、EASE協調フィルタリング法（EASE:CF法）を提案している。この手法を用いた場合、全体の60％以上が欠損したデータを用いても、高い精度で予測できることが確認された。

EASE：CF法の基礎である協調フィルタリングは、情報検索の分野で生まれた予測手法である。著名な適用例として、顧客に「お薦め書籍」を提示するAmazon.com社の書籍推薦システム（http://www.amazon.co.jp）が挙げられる。このシステムの推薦がかなり的を射たものであることは有名だが、使われているデータの大部分が欠損していることは知られていない。

このシステムは、各顧客が読了した書籍を「5（好き）」～「1（嫌い）」の5段階で評価したデータを蓄積している。しかし、読了していないなどの理由から、評価が入っていないデータ（欠損データ）も多い。予測のベースとなるデータがこのような虫食い状態でも、書籍推薦システムは協調フィルタリングにより、かなり正確に顧客の好みを予測できるのである（**図表6-9**）。

協調フィルタリングは、データ中の類似した事例から予測対象を類推する手法である。これをプロジェクトの特性予測に用いた場合、経験を積んだプロジェクト・マネージャが行うような類推プロセスを自動化できる。ベテランのマネージャは、過去に自分が関わった案件から、現行プロジェクトと似たものを思い出し、その経験を基に予測する。情報サービス産業協会（JISA）が2004年度に実施した「情報サービス産業における受注ソフトウェア開発の技術課題に関わるアンケート調査」によると、全体の4分の1以上のプロジェクト・マネージャが、予測の根拠として「類推」を挙げている。

図表6-10に協調フィルタリングを用いたプロジェクト特性の予測手順を示す。図中に示した各ステップでは次のような処理を実行する。ステップ1では、プロジェクト間の類似度を計算する。各変数の値が取り得る範囲を正規化したデータを用いて、現行プロジェクトと過去プロジェクト間の類似度を計算する。類似度の計算方法は、コサイン類似度、相関係数、ユークリッド距離など、8種類のアルゴリズムから予測に適したものを選択して用いる。

ステップ2では予測値を計算する。類似度の高い順に上位k個（図表6-10の

図表6-10●協調フィルタリングによるプロジェクト特性の予測

		開発言語	開発種別	概算FP	要員数	開発工数
類似度: −1.0	過去プロジェクトA	データ欠損	0（保守）	100	40	250
類似度: +1.0	過去プロジェクトB	0（Java）	1（新規）	データ欠損	8	40
類似度: +0.9	過去プロジェクトC	0（Java）	データ欠損	25	6	35
	ステップ1：類似度を計算する					
	ステップ2：類似度で加重平均するなどして値を予測する					
	現行プロジェクトX	0（Java）	1（新規）	40	10	予測値:37

例ではk=2）を類似プロジェクトとして選び、それらのデータを加重平均して現行プロジェクトの特性を予測する。図表6-10の例では開発工数を予測している。予測値の計算方法は、単純加重平均、増幅加重平均など、10種類のアルゴリズムから予測に適したものを選択する。

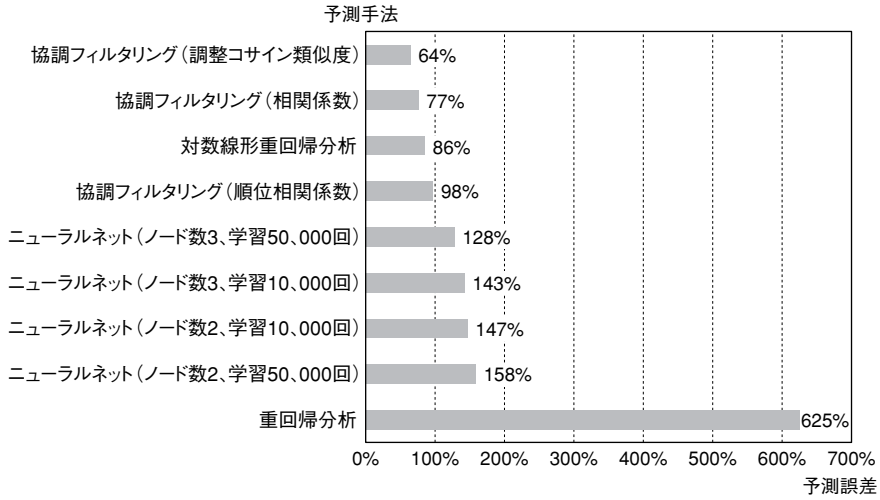
協調フィルタリングを実施する際、各ステップでデータ欠損の大きさや分布状況に応じて柔軟に予測方法を変更できる。これにより、欠損部分が大きくなっても安定した精度を発揮できる。アルゴリズムの選択も、次の手順によって最適化できる。まず、過去プロジェクトのデータに入っている既知の値を一つ隠してしまう。次に、各アルゴリズムを用いて隠した値を予測してみる。これを繰り返し、隠した値と予測値の間の誤差が最も小さくなるアルゴリズムを選択すればよい。

6.5.3 SECにおける適用事例

EASEプロジェクトでは多くの企業の協力のもと、SECが収集したデータに対してEASE：CF法を適用し、工数予測の精度を評価した。SECは2005年3月時点で、国内ソフトウェア開発会社15社が実施した1009のプロジェクトについて、開発工数や工期など約490種のデータを収集した。特に重要な「工数」や「規模」などのデータには欠損が少ないものの、15社のデータ収集ポリシーが少しずつ異なるため、全体としては87.3%の部分が欠損していた。

このように欠損の多いデータを使った場合の予測精度の結果を**図表6-11**に示す。協調フィルタリング（類似度計算アルゴリズムとして「調整コサイン類似度」を使用）の精度が最も高く、重回帰分析の精度が最も低いことが分かった。他の手法については、統計的

図表6-11●SEC データを用いた各予測手法の評価結果



な検定を行うと精度に有意差がないことが分かった。

この事例では、協調フィルタリングを用いても予測誤差は64％となった。これは、開発の特徴や状況が異なる複数企業から収集したデータを、ひとくくりに扱った点が原因である。単独企業から収集したデータを用いれば、精度を高められる可能性が極めて大きい。例えば、ある企業と共同で行った事例では、データ量が少ないにもかかわらず予測誤差を28％まで低減できた。

EASEプロジェクトでは、協調フィルタリング実行エンジンをはじめ、プロジェクト類似性可視化ツールや、予測自動化ツールなどを開発しており、企業との共同研究も盛んに行っている。本手法の詳細、手法適用のためのツールについてはEASEプロジェクトのWebサイト（http://www.empirical.jp）を参照してもらいたい。

6.6 失敗事例の分析とその傾向

見える化手法を検討するに当たり、プロジェクトの失敗事例や経験豊富なプロジェクト・マネージャの知見を集め、分析を行った。より現実的な問題を調べることによって、実践的な手法を検討するためである。そして、プロジェクトの問題がどのような部分にどのような形で現れるのかを検討し、分類のためのフレームワークとして開発した

のが「症例分類表群」である。この症例分類表に対して、過去の失敗事例やヒアリングシートの各項目がどのように関連するのかを調べたことで分かったことを説明する。

まず、SECが収集した過去の失敗事例（77件）の「症例」を分類していったところ、①「スキル・知識/経験・手順・文化」の不足・不在、②「モノ・金」の不足・不備、③「人」の疲労・疲弊・離脱、にかかわるプロジェクトの問題が多かった。この分類結果だけでプロジェクトの問題との関連性を論じることはできないが、例えばプロジェクト・メンバーのスキル・文化の不足によってプロダクトの品質が下がり、その結果キーパーソンへの負荷集中やメンバーの疲労、プロジェクト・マネージャの交代・離脱を誘発するという問題の連鎖が垣間見える。まさしく「人に始まり、人に終わる」というプロジェクト活動の本質が見えてくる。

このように実事例を検討してきた症例分類表に対して、今回作成したヒアリングシートの項目を症例分類表にマッピングしたところ、実事例のマッピングと同様のマッピング結果が得られた。これは、作成したヒアリングシートが、ほぼ現実のプロジェクトの問題をうまく捉えられる内容であることが検証できた。

さらに詳細に調べると、ヒアリングシートのチェック項目によって、多数の症例パターンに該当するものもあれば、特定の症例と結びつくものもあった。特に、多数の症例パターンに該当するチェック項目の内容には注意が必要だと言える。そういう要注意のチェック項目のいくつかを**図表6-12**に示す。

次に、1つの症例パターンが関連するヒアリングシートのチェック項目数を調べた。症例パターンは、複数のチェック項目と関連付いているが、これはヒアリングシートの質問内容がいくつかの症例パターンの候補を示そうとしているためだ。つまり、数多くのチェック項目と関連付いている症例パターンは、専門家の重大な関心事であると考えられる。

集計したところ、「ルール・手順の不足」、「コミュニケーションの不足」、「ルール・手順の不備」、「理解・合意の不足」という症例パターンが多数のチェック項目と関連付けられていた。主に、ルール・手順がきちんと整理・実施されているかどうか、コミュニケーションに問題ないかどうかが、プロジェクトの問題発生パターンの一つであることが分かる。

図表6-12●複数の症例パターンに関係するヒアリングシート項目上位4位

ヒアリングシートの質問内容	症例パターンと予想
（チェック項目番号58）実施責任者（プロジェクト・マネージャ、プロジェクト・リーダーなど）および有識者を交え、協力会社に依頼する作業が適切な規模、難易度であることを検討しているか？	・「協力・関係会社」の不足・不在・不備 ・「技術」の不足・不在・不備 ・「見積もり」の不足・不在・不備 ・「知識・経験」の不足・不在・不備 ・「検討」の不足・不在・不備 ・「計画」の不足・不在・不備 このチェック項目に問題があると、技術力・見積もりなど、協力会社選定上の問題や、見積もりミスが引き起こす計画上の問題など、問題が多岐にわたりやすいことが分かる
（チェック項目番号70）レスポンスの確保や無応答状態をなくすなどの方式設計については十分に実施しているか？	・プログラムの不足・不備 ・仕様の不足・不備 ・ハードウェアの不足・不備 ・ツールの不足・不在・不備 ・設計の不足・不備 ・テストの不足・不在・不備 ・検討の不足・不在・不備 このチェック項目に問題があると、方式設計の不足によって、プログラムをはじめとするプロダクトの品質・方式に問題があったり、必要なツールの不足に陥ったり、設計・テストでもれが生じたりすることが分かる
（チェック項目番号51）スケジュール・作業内容について顧客との認識（作業レベルの意識）にズレがないか？	・ドキュメントの不足・不在・不備 ・レビュー・検証の不足・不在・不備 ・理解・合意の不足・不在・不備 ・スケジュールの不足・不在・不備 ・範囲・役割の不足・不備 このチェック項目に問題があると、作業実施上に必要なドキュメントに不足が生じたり、顧客とのレビュー・検証で相互の認識にずれがあったり、役割・範囲があいまいになってトラブルになり得ることが分かる
（チェック項目番号71）トラフィック量とトラフィックパターンについての想定の検討は十分に行われているか。どのような対応策をとるか明確になっているか？	・プログラムの不足・不備 ・仕様の不足・不備 ・ハードウェアの不足・不備 ・ツールの不足・不在・不備 ・設計の不足・不備 ・テストの不足・不在・不備 このチェック項目に問題があると、システムのキャパシティ・プラニングに問題があることになり、顧客が求めるサービス品質を満足できない可能性がある。具体的にはプログラムや仕様の設計品質、テスト品質、ツールの不備・不足という現象で現れてくる

おわりに

「見えないものを見えるようにしたい」、「見えたものを見えたまま知らせたい」、そして「問題があれば改善したい」――。

本書はこの考え方のもとに、一貫して失敗しそうなプロジェクトを救うための「見える化」手法を説明してきた。特に今回は、システム開発の下流工程をターゲットとしている。これは「見える化」を説明しやすいためである。システム開発の下流工程では、それまでの工程に起因する品質不良の問題が顕在化しやすいという特徴がある。また、下流工程で問題が発生した場合は、解決に当てられる時間や手段が限定されている。対処方法・改善方法への道筋を決めやすいのだ。

改善手法における取り組み

プロジェクトの過去の失敗事例に対して、SECは症例分類や対症療法について検討してきた。プロジェクトの失敗事例を分類し、それらへの対策を人間の英知として蓄積していくことにより、プロジェクトで起こった問題に対し、より良い対応ができるようになる。まさしく医療の世界のように、プロジェクトの病理学が確立され、症例に対する処置や処方が議論され、より良い治療が講じられ、さらに病気にならないための予防策も考え出されるようになるだろう。また、失敗事例を分析することで、問題への対処方法や予防策について、体系的な方法論が研究できるようになるだろう。

SECは今後、次のようなことを検討していきたい。

- 失敗事例の収集と分析
- 対症療法および再発防止策の実践方法
- 対策の留意事項

今後の取り組み

本書を中期的なマイルストーンとして、SECは今後2年をかけ、プロジェクトの見える化手法を体系化する（**図表A**）。今年度は下流工程のチェックシートなどを作成したが、来年度以降は、上流工程および中流工程のチェックシートなどを作成する。これらの成果をまとめて、プロジェクトの全工程で利用可能な見える化手法を確立する考えである。

来年度は、上・中流工程の「見える化」を実施する予定である。システム開発プロジェクトは「見切り発車が日常的に行われている」という仮定に立ち、以下の項目を検討する。ここで言う「見切

り発車」とは、契約前に設計作業に着手したり、開発要員がまだ足りない状態なのに、開発作業に着手したりすることである。

- 見切り発車状況を乗り切るスーパープロジェクト・マネージャの知見を形式知にする
- 見切り発車を想定して、その状況を定性的・定量的に見える化する
- 上記を手法化し、上・中流工程におけるプロジェクトの見える化手法を確立する

優れたプロジェクト・マネージャはリスクを予知し、ネガティブ・インパクトを最小限に抑える。この「プロジェクトを見える化するノウハウ」を広く普及させていくことが、日本のIT産業の発展に貢献するものと信じている。

図表A●本部会の活動予定

	2005年度	2006年度	2007年度
研究活動の狙い	・下流工程のプロジェクトの見える化方法整備	・上中流工程でのプロジェクトの見える化方法整備	・見える化の体系化
成果物	・見える化チェックリスト ・対応策、手法 ・解説書	・見える化チェックリスト ・対応策、手法 ・解説書	・解説書

付録

見える化のツールと関連資料

1．管理帳票について

プロジェクトの状況把握に使う最も基本的なツールとして、管理帳票が広く用いられている。組織によって名称や項目には違いがあるが、レビュー記録、問題記述票、バグ票、進捗管理表などと呼ばれる帳票類を指す。

これらの帳票に記入される項目や、帳票を管理する管理表の項目から、プロジェクトに内在する問題を明らかにし、プロジェクト・マネジメントに活用する。ITプロジェクトの管理データに関しては、日本規格協会 情報技術標準化研究センター（Information Technology Research and Standardization Center：INSTAC）の「ソフトウェアCALSに関する調査研究委員会WG4（管理データデータ交換）」の報告書（1997年3月）が参考になる。

また、1996年10月から情報処理事業振興協会（IPA）の企業間高度電子商取引推進事業の一環として、ソフトウェアCALS環境の構築と実証実験が行われた。ソフトウェア開発・保守フェーズでの企業間のプロジェクト・マネジメント情報を共有・交換する「管理データ交換支援機能」を開発し、実証実験に用いた。この実証実験では、「障害連絡・回答票」、「問題連絡・回答票」、「レビュー指摘連絡・回答票」と「仕様変更依頼・回答票」を管理データとし、その形式と内容表現をルール化した作成要領を整備している。情報交換において、情報の形式規定、内容規定（ルール化）は、対象文書の精度向上、責任の明確化、信頼性の確保には必須である。また、連絡から回答までの一連の手順も共有し、対象文書のステータスを管理する。これにより対応状況をフォローアップでき、プロジェクトの進捗管理、品質管理に有効であったとの報告もされている。

なお、実証実験の成果は、1998年のソフトウェアCALS・EXPOで展示発表され、ドキュメントやツール群として結実しているので、参考にしてもらいたい。

2．チェックシート

優秀なプロジェクト・マネージャの経験則から導いた数十個のチェック項目により、プロジェクトに問題が起こっているかどうかを定性的に把握するチェックシートを開発した。プロジェクト・マネージャなどが自己診察に用いる「自己評価シート」と専門家チーム（例えばPMO）によるヒアリング診断で用いる「ヒアリングシート」の2種類がある。自己評価シートは116ページを、ヒアリングシートは122ページを参照してもらいたい。

3．プロジェクトにおける問題事象と対策事例

開発現場の有識者から失敗事例で現れた事象や原因、対策を聞き取り、失敗事例データベースを作成した。具体的な失敗事例とその対応策をあらかじめ知識として身に着けておけば、プロジェクト遂行中に発生する様々な問題に対して適切な処置を実施することが可能になる。特に経験の少ないプロジェクト・マネージャにとっては有用な知識となる。有識者の選定は、プロジェクト・マネージャ10年以上、あるいは品質保証業務でプロジェクトの火消し経験10年以上を基準に行った。失敗事例データベースは142ページを参照してもらいたい。

4．測定項目リスト

プロジェクトの状況を定量的に見える化するためには、どのような測定項目が必要なのかを決定する必要がある。測定項目を定める際には、その測定目的を明確にし、測定にもれがないよう、網羅的かつ体系的な測定項目リストが必要である。有識者の知見や世の参考文献を基に、「測定項目リスト」として体系化した。174ページを参照してもらいたい。

5．EPMツールの分析指標

測定項目リストには多数の測定項目がリストアップされているが、このうちのいくつかは、EASEプロジェクトで開発したEPMのようなプロジェクト・モニタリング・ツールで自動的に測定し、グラフ化して表示できる。192ページの一覧表は、EPMで測定・分析可能な項目をまとめたものである。

6．症例分類表

プロジェクトで起こる問題の「発生個所」と「状態」に注目し、過去のプロジェクト失敗事例を分析して「症例分類表」を体系化した。症例分類表では、問題の発生個所とその状態を表の縦軸と横軸に取り、それぞれに対応す

る症例を分類したマトリックスである。

症例分類表は4種類ある。まず、縦軸の「問題発生個所」と横軸の「状態」の組み合わせから、どのような症例パターンがあるのかを一覧したのが「症例分類表（一般マップ）」（198ページ参照）である。縦軸と横軸の構造は同じだが、中身をヒアリングシート、測定項目リスト、失敗事例データベースの項目と関連付けたものが、「症例分類表（ヒアリングマップ）」、「症例分類表（測定項目マップ）」、「症例分類表（事例マップ）」である。

「症例分類表（ヒアリングマップ）」はヒアリングシートのチェック項目番号とリンクしている（202ページ参照）。「症例分類表（測定項目マップ）」は、測定項目リストの測定項目番号と関連付けている（206ページ参照）。「症例分類表（事例マップ）」は、事例番号と結び付けている（208ページ参照）。

7. 知識エリアについて

本書では、PMBOKの9つの知識エリアに6項目を加えて、15の知識エリアを定義した。PMBOKに追加したの

図表A●拡張知識エリアの定義

拡張知識エリア	定義
顧客	プロジェクトの利害関係者のうち、システムの仕様及び予算について最終決定権を持っている人もしくは組織のこと。受託型のITプロジェクトでは、顧客と協働で、最終的な成果物であるシステムを作るための合意形成を行っていく場合が多い
組織	システム開発に携わる組織のこと。ITプロジェクトでは、多段階の下請け構造を取る場合や複数の開発会社が参画するマルチベンダー方式が取られることもある。現実的には、開発組織構造によって、『人的資源』や『調達』のあり方に様々な制約が生じることになる
基本動作	システム開発における常識及び開発者が身に付けておくべき当たり前の動作のこと。システム開発マネジメントの内容も含まれる
モチベーション	システム開発に携わる人のやる気、動機付けのこと。システム開発に携わる人の心理的側面、労務環境や個々人の成長目標などのキャリアディベロップメントに関する事項も広く含まれる
技術	システム構築技術に関するマネジメント項目。システム開発マネジメントの内容であり、PMBOK™には含まれない業種固有のマネジメント領域に当る
課題管理	ITプロジェクトにおける課題管理に関わるマネジメント項目。PMBOK™では、監視・コントロールプロセスのマネジメントに該当するが、本書では、プロジェクトの下流工程に焦点を当てているため、特に重要な領域として拡張エリアに追加している。システム開発の現場で当然行なわれるべき課題管理のあり方を扱う

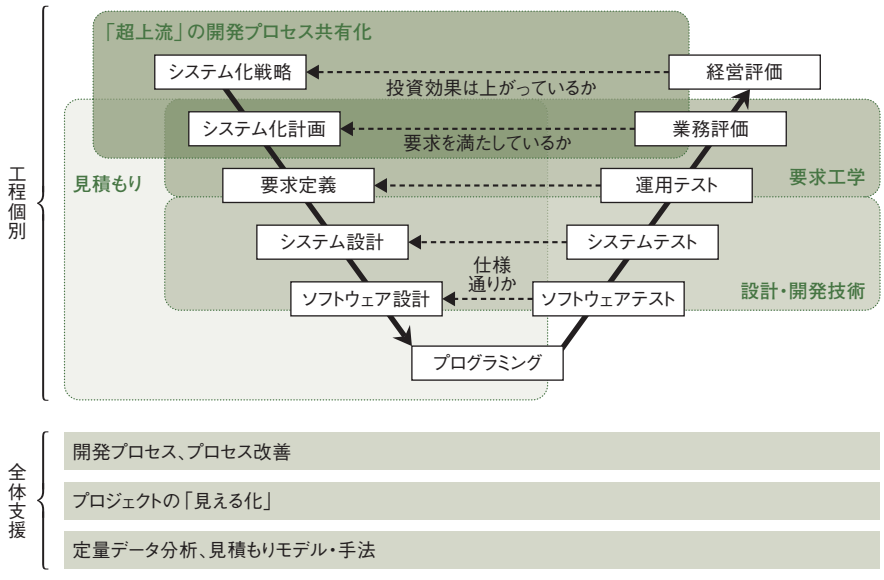
は、ヒューマンウェアの観点から、『顧客』、『組織』、『基本動作』、『モチベーション』、PMBOKには含まれないITプロジェクト固有のマネジメント領域である『技術』、そしてプロジェクトの下流工程における監視コントロール・プロセスに注目した『課題管理』である。

その6つの拡張知識エリア（**図表A**）についてそれぞれ説明する。

8. SECの活動と「プロジェクト見える化部会」について

SECの「プロジェクト見える化部会」は、SECエンタプライズ系活動の中で、プロジェクト開発工程の全体支援に位置付けられている（**図表B**）。本部会は、2005年度に活動を開始し、3カ年計画で研究を進めている。初年度の2005年度は、下流工程に着目して見える化手法を整備した。来年度以降は、上・中流工程の見える化手法を整備する予定である。

図表B●SECエンタプライズ系活動



付録 2. チェックシート【自己評価シート】

No.	知識エリア	チェック項目	評価基準	マネジメントにおけるヒント	対策案
S1	統合	プロジェクト計画が作成され、レビューされているか？	プロジェクト計画書は、作業着手から規定期限以内に作成され、レビューされていること	期限が設定されていない場合は、作成期限を設定すること	プロジェクト計画が現状と大きく乖離している場合は、計画が立っていないことと同意であるので、時間がなくてもプロジェクト計画を作成・更新し、利害関係者からの同意を得ること
S2	統合	構成管理の対象として、納入物件、必要な作業、成果物が明確となっているか？	顧客との間で成果物のイメージが明確になっていること	顧客との間で成果物のイメージが明確でないとその後の作業の割り振りや規模の見積もりなどで大きな判断ミスをする可能性がある	構成管理対象が明確になっていない場合、構成対象を明確にした上で、構成管理ルールを作成する
S3	統合	すべての作業項目に対してスケジュール化しているか？	スケジュールに記載のない作業項目がないこと	クリティカルパスを明確にすることによって、遅れてはならない作業の重点監視を行えるようになる	プロジェクト運営上のマイルストーンである ・テスト工程 ・運用教育 ・移行作業 ・本番リリース ・立会い予定 ・クロージング作業 などをはじめ、顧客主導のイベントである ・ビル停電 ・重要な会議 ・他ベンダーとの結合テスト などをヒアリングして、クリティカルパスを明確にする
S4	統合	予定外の作業が頻繁に発生しているか？	スケジュールに記載のない予定外の作業が発生していないこと	計画が頻繁に変更されると、プロジェクトに混乱が生じやすい。プロジェクトが混乱した場合には、計画の実質的な見直しと検討が必要である	・作業効率を考慮して、作業項目を統制する必要がある ・顧客、協力会社、プロジェクト・マネージャなどステークホルダー全員で予定表にない作業についてヒアリングを実施し、作業項目としてリストアップする ・その上で、作業の優先順位付けや作業順序の検討、取捨選択を実施し、全体的な効率化を検討する
S5	統合	重要度の高いミッションクリティカルな機能を提示し、管理しているか？	プロジェクトの重要機能の優先順位や、留意すべき項目が挙げられていること	ミッションクリティカル性の高い機能要件は失敗してはならない機能なので、計画・設計時から入念に管理され、実行され、監視される必要がある	・クリティカルパスを明確にする ・その上で、特に品質の悪いプログラムに対する管理、テスト、レビューを重点的に実施する ・クリティカルパスについては特に毎日の進捗報告を義務付け、管理を徹底する ・優秀なエンジニアを投入する
S6	統合	全体テスト方針と役割分担の検討は十分か？	全体テスト方針（何をどのテストで確認するか）と役割分担を記述したドキュメントを作成し、関係者とレビューしていること	検討が十分でない場合、過去事例を参考に全体テスト方針（何をどのテストで確認するか）を整理し、ステークホルダーの役割分担を決める	・単体テスト、結合テスト、総合テストのそれぞれのスケジュールと品質目標が未計画の場合は過去事例などを基に整理し、計画する ・テスト体制も明確にする ・特に結合テスト以降は優秀な人材（各チームのキーパーソンや優秀なエンジニア）を投入し、テスト工程が遅れないような体制を組む
S7	統合	進捗状況を定期的に確認しているか？	例えば、以下のような項目について進捗を把握していること ・計画の変更数 ・実施レビュー数と種類 ・日程の予実績の差異 ・コーディングステップ数の予実績 ・コードインスペクション網羅率 ・リスク数（定義数、監視数、軽減数）	進捗状況は協力会社からの進捗報告をうのみにせず、現物確認で実施すべきである	・定例会議ができていない理由を探る - 必要性の理解 - 時間の確保 - 複数社のメンバー - 場所の問題 ・定例進捗会議を設置する - プロジェクトの期間とステータスに応じて周期を決める - 関係者ではなく責任者をメンバーにする - 階層化を図る ・設置不能な場合は代替策を講じる - 忙しいときは時間をずらす（場合によっては遅い時間でも）、短時間ですます - 地理的原因なら、TV 会議、電話会議などの工夫

No.	知識エリア	チェック項目	評価基準	マネジメントにおけるヒント	対策案
S8	統合	構成管理が行なわれているか？	文書化された構成管理の手順やルールに基づいて、構成管理対象への変更を記録し、報告させる運営が行なわれていること	実際の構成管理が、手順やルールに基づいて記録され、報告させる運営になっていることを確認する	・構成管理ルールがある場合には、ルールに基づく運営を徹底する ・構成管理をしていない場合には、その原因を追求する - 構成管理の必要性を理解しているか？ - 構成管理をしないと何が起きるかを考えてみる - 言葉に惑わされないで、何をすることかを理解する ・担当を決める（厳格な人が良い） ・構成管理で行うべき事項・ツールを検討する ・原本をキッチリ押さえること
S9	スコープ	顧客の要件とシステムにおける機能、性能が一致しているかどうか確認しているか？	顧客の要件を開発側がきちんと理解していること	顧客の要件から大きくずれている状況は問題である	・要件定義書がある場合は、現状と要件定義書を比較して特に乖離の大きい部分があればその部分の状況を報告し、顧客側と認識を合わせておく必要がある ・要件定義書には記述されていない要件がある場合、早めに洗い出し、クリティカルな要件になるようであれば実装を機能面・費用面から検討する ・要件変更要求が頻繁にある機能については、開発を一度止めて、議論し直して仕様を確定させる必要がある
S10	スコープ	要件定義と仕様変更要望一覧間の関連付けを明確にしているか？	顧客側のニーズ（要求）と要件定義書の項目の関係性を明確にしていること	要件定義と仕様変更の対応付けを管理することによって、顧客要件の合致性を確認する	・要件定義と仕様変更との関連付けを明確にする ・要件定義と仕様を対応表にして、相互に影響する ・要件定義変更や仕様変更を管理する
S11	スコープ	仕様変更が一覧表などにより管理され、変更履歴が残されているか？	仕様変更管理表などで管理していること	以下の項目について確認する ・仕様変更管理表の所在（どこにあるか） ・仕様変更管理表の記入者・保管責任者 ・仕様書の変更履歴と仕様管理表が一致していること	・要件定義と関連付けを持っている仕様変更一覧表を作成する
S12	スコープ	仕様として設定された要求の実現可能性に問題がないことを確認したか？	実現可能性を技術面、性能面から検証していること	計画または設計時に実現可能性そのものについて検討を行っているかどうかが重要である。実現可能と判断するには、例えばコスト、機能、納期に対してリスクを認識し、その対応策が見えている必要がある	・実現可能性を検討していない場合は早い段階で検討しておく ・下流において大半の実装が終わっている場合は、実現可能性をすでに見極めているかもしれないが、それらも含めて全体を俯瞰できるよう、要求された項目の実現可能性の一覧を作成する ・実現可能性が低いと思われる機能がすでにテストに入っている場合は、品質に問題がないかを特に注意すること
S13	タイム	前工程の作業が完了し、プロジェクト内（プロジェクト・マネージャ）の承認を得ているか？	前工程に積み残し課題がある場合は、解決のメドをつけた上で次工程に引き継いでいること	上流工程の品質の悪さは下流工程でより大きな悪影響を及ぼすことがある	・品質状況を確認して、問題がある場合は品質向上策をとること
S14	タイム	前工程は、品質保証部門による確認が完了し、合格となっているか？	条件付きの合格の場合や改善点を指摘された場合は、課題管理の対象としていること	以下の項目について確認する ・工程区切り基準があるか？ ・各工程でやるべきことをすべて消化できているか？ もし、積み残しがあれば積み残した項目を明確にして進め、次の工程で解消させる	・品質状況を確認して、問題がある場合は品質向上策をとること

No.	知識エリア	チェック項目	評価基準	マネジメントにおけるヒント	対策案
S15	タイム	テスト計画について、実施責任者(プロジェクト・マネージャ、プロジェクト・リーダーなど)を含めたレビューにて内容を確認しているか?	計画の妥当性を検討していること	テストによっては、顧客の確認が必要な場合もある	・テスト計画について特にリーダー、協力会社の認識にずれがないかを確認し、問題がある場合は問題点の抽出と計画の見直しを実施する
S16	タイム	クリティカルパスが明確になっているか?	プロジェクトの根幹にかかわる機能や成果物に関する計画はどのパスであるかを認識し、管理していること	スケジュール遅延に直結するクリティカルパス上の作業については、意識的にほかよりも重点的にチェックする必要がある	・キーパーソンを集めてクリティカルパスの抽出作業を行う。その上で、クリティカルパスに関連する作業は特別体制を敷く
S17	タイム	他のステークホルダーからの提供物のスケジュールが守られているか?	提供物のスケジュール状況を継続的にウォッチし、守られなかった場合の影響度に応じて、事前に対策を立てていること	他のステークホルダーの役割と提出物のスケジュールを別途マネジメントすることにより、役割分担があいまいになっていることを原因とするプロジェクトの遅延を防止できる	・品質状況を確認して、問題がある場合は品質向上策をとること
S18	コスト	予実績管理は適切に実行されているか?	収支管理計画を事前に定義していること(週次 / 月次の実績把握)	・期間(工程の開始/終了など) ・要員の山積み ・WBS について予実績管理を行なう	・予実績情報を収集・整理して、今後の計画の見直しを実施する ・すべての作業の予実績情報の収集が困難な場合は、クリティカルパスについての予実績だけでも集めて評価すること
S19	コスト	追加コストが発生しないか?	必要な業務・知識領域と要員のスキルの比較を行ない、不足している業務・知識領域について、コンサルタントや専門家のアサインを考慮していること	システム基盤に問題が発生した場合はスペシャリストのアサインが有効である。また、特殊な業務に関する知識が不足している場合は、外部の専門会社などに依頼することを考慮しておく必要がある	・最適なスペシャリストを投入する計画を立てる
S20	品質	日々のテスト項目消化件数、不良摘出数、不良対策件数を入力する仕掛けが整備されており、バグ曲線による管理が行われているか?	テストの進捗と結果を管理する仕組みができていること	・以下の帳票類があること(現物で確認) - 記入済みのバグ票 - 現時点までのバグ曲線(あることが望ましい) - テスト実施要領 - テスト実施・結果の管理票 ・集計の仕組みと責任者が明確になっていること(専用ツールを使っている/Excel などのツールで集計/人手集計) ・毎朝会議を実施していること(結合テスト以降)	・テスト実績管理ツール(Excelの表やDBなど)がない場合は早急に準備する ・すでにテスト工程に入ってバラバラにテストしている場合は、実績管理を一元化し、すでに終わった実績情報を収集・管理する
S21	人的資源	ステークホルダーの見極めができているか?	ステークホルダー分析をきちんと行い、交渉窓口を明確にしていること	誰がそのプロジェクトのキーパーソンであるかを明確にする。そのために必要なものがステークホルダー図である。関係している人の名前だけでなく、財布を握っている人、仕様を握っている人を明確にして、一番効果のある働きかけを行なうことが肝要である	・ステークホルダー分析を行い、誰がキーパーソンなのかを明らかにする。形式的な体制図のほかに、実質上のキーパーソンを押さえることが重要である

No.	知識エリア	チェック項目	評価基準	マネジメントにおけるヒント	対策案
S22	人的資源	要員の手配（量的）はできているか？	工程の変わり目など、体制が増員されるタイミングに間に合うように要員の手配がされていること	担当者ごとに作業開始時期、終了時期を明確にして作業をアサインしていること。 また、前工程における仕様変更などで必要な人員の増減が必要になった場合には、工程の終了を待たずに速やかに手配する必要がある	・山積みではなく要員遷移分析を行い、WBS ごとに、必要なスキルを持った要員の過不足を明らかにする ・前工程の終了前に、プロジェクト・マネージャがどれだけ次工程の要員手配に注力するかが重要である ・要員の手配は、早めに、過大気味に行う
S23	人的資源	求められる業務知識のキーパーソンを獲得できているか？	求められる業務分野の経験者を押さえていること	漠然と「業務分野の経験者」と捉えずに、求められる経験・知識レベルをしっかり把握する必要がある	・求められる経験・知識レベルを具体的に把握する ・次に、業務知識のキーパーソンを確保できているかを確認する ・社内に人材がいるときは、相談やチェックだけでも参加してもらう交渉をする ・社内に人材がいないときは、 - 協力会社から調達する - 理解力のある人材を起用または採用する - プロジェクトから降りる（失敗コストと罰金 / 信用失墜との見合い） ・業務をよく知っていても引っ込み思案やリーダーシップが欠如していれば、キーパーソンにはなれない
S24	人的資源	求められる技術スキルのキーパーソンを獲得できているか？	パッケージ使用の場合、フィット&ギャップ分析ができる要員やプロジェクトで使用するソフト（DB、ミドルウェア、ツールなど）の経験者がキーパーソンとなっていること	対象プロジェクトのキーとなる技術は何かを理解する（求められる技術スキルはプロジェクトによって異なる）。 該当技術のキーパーソンの経歴・レベルを確認する（質問する人の技術レベルにも依存するので注意。自分の技術レベルが低いと他人を「高い」と言う傾向あり）	・求められる技術レベルを具体的に把握する ・次に、技術のキーパーソンを確保できているかを確認する ・社内に人材がいるときは、相談やチェックだけでも参加してもらう交渉をする ・社内に人材がいないときは、 - 協力会社から調達する - 理解力のある人材を起用または採用する - プロジェクトから降りる（失敗コストと罰金 / 信用失墜との見合い） ・技術をよく知っていても引っ込み思案やリーダーシップが欠如していれば、キーパーソンにはなれない
S25	コミュニケーション	顧客階層別、チーム間、協力会社との会議体が決められ実行されているか？	ステークホルダーと協議し、必要な会議体を設定していること	非公式なコミュニケーションパスも準備しておく。また計画だけではなく、実質的にその会議体が機能していることが重要である	・顧客、協力会社など、プロジェクトの状況を見て必要な会議体を設定し、定例化する
S26	コミュニケーション	顧客への報告は随時行っているか？	顧客との会議体を決めて、実際に行なわれていること	トラブル対応や調査依頼に対しての中間報告を行い、随時情報を提供する必要がある	・課題やリスク、プロジェクト状況などを報告するための会議体を設定する
S27	コミュニケーション	プロジェクト計画書は、全員が参照できるようになっており、全プロジェクト・メンバーへ周知しているか？	プロジェクト計画書のありかを全員に連絡していること	各メンバーがプロジェクト計画書の内容をきちんと理解していることが重要	・プロジェクト計画書を最新に保つようにし、特に重要な項目については必ず目を通すように徹底する ・特に新しくプロジェクトに参加したメンバーに対して説明することを忘れないようにする
S28	リスク	リスク分析により、リスク項目を明確にしているか？	リスク管理表などを起こして管理していること	リスクの洗い出しには、ステークホルダーが集まって討議することが有用	・チーム・メンバーを集めてリスクの洗い出しを行い、洗い出したリスクが発生する確率と発生した場合の影響を判定する
S29	リスク	リスク対応策を実施するタイミングが明確か？	リスク管理表に明記されていること	リスクの予防対策には、回避、転嫁、軽減などがある	・洗い出したリスクごとにリスク対応策を検討し決定する。さらに、リスク担当者を決めて、対応策を実施する時期をウォッチングさせる

No.	知識エリア	チェック項目	評価基準	マネジメントにおけるヒント	対策案
S30	リスク	コンティンジェンシー（予備予算・予備日）を持っているか？	リスク管理表に明記されていること	発生時の対策には、コスト予備、スケジュール予備の確保、要員の代替や追加投入が必要となる	・コンティンジェンシー（予備予算・予備日）の追加を顧客と上級管理者に依頼する
S31	調達	協力会社の作業と成果物を定期的に監視しているか？	定期的に確認していること	確認時にはその報告や成果物の内容チェックまできちんと実施していることが重要	・協力会社に協力してもらい、作業報告とともに成果物（特に仕様書などのドキュメント）を最新の状態にするように管理する ・特にソースやシステム環境については構成管理チームを作って徹底的に管理する
S32	顧客	顧客のプロジェクト目的は明確か？	プロジェクト計画書に開発側の目的だけでなく、顧客の目的も記載していること	顧客との交渉を円滑に進めるために、顧客がシステム開発を通して、何を実現したいのかを把握しておくことが重要	・不明確な場合には、顧客にヒアリングし、その結果を文書化しておく
S33	技術	方式要件（負荷・性能・信頼性）と方式設計結果は妥当か？	方式要件と方式設計結果に関するレビューを実施していること	顧客に責任を持って方式要件を決めてもらい、それに対する設計を行う。有識者に要件と設計結果の妥当性をレビューしてもらう	・方式検討を実施していないにもかかわらず、プログラミング工程、テスト工程に入っている場合は、仕様上の問題が発生する可能性があるので、特にクリティカルパスについては方式の問題がないかを再確認する
S34	組織	メンバーのミッションが明確で、各人が意識しているか？	プロジェクト・マネージャと各メンバーとが各人のミッションを共有していること	・各人の責任分担を明確にする ・いつまでに誰が何を行なわなければならないかを確定することが重要である ・これがあいまいだと「言ったのに」「そうじゃなかったのに」というプロジェクト・マネージャの失敗台詞を聞くことになる ・プロジェクト・マネージャは部下に責任の範囲やいつまでに何を、ということを言葉ではなくて、計画として説明する必要がある	・必要なスキルを明らかにして、外部研修を含む適切な教育を行う ・協力会社から社員代替要員を確保する
S35	基本動作	基本となる動作（しつけ・作法）が徹底されているか？	例えば「打ち合わせを行うときは議事録をとること」、「設計内容はドキュメント化すること」など、基本的なしつけ・作法について全メンバーが問題なく認識していること	チーム全体が「多少手抜きをしても大丈夫」という雰囲気にならないよう、決められたことはきちんと守ることを徹底して指示する。コミュニケーションを良くすることと「なあなあで済ます」ことは別であり、緊張感を持って仕事に取り組むことが大切である	・基本動作についての教育的な指導を普段から心がけるようにする
S36	基本動作	部下の状況を把握しているか？	公式、非公式のコミュニケーションを部下と行なっていること	部下から何の報告もない場合は、報告しづらい問題を抱えている場合もあれば、話しづらい雰囲気などコミュニケーション上の問題点があることがある。会議ではないところでも、コミュニケーションを随時行っていることが大切である	直接部下に状況をヒアリングしたり、部下とかかわりのある第三者にヒアリングして部下の状況を多面的に把握する
S37	モチベーション	プロジェクト計画に対してメンバーの気持ちがしらけていないか？	プロジェクトの日程計画や実現可能性などに関してメンバーが不信感を持っていないこと	必要な作業を積み上げた計画になっており、スケジュール、コストのベースラインは、プロジェクト・メンバーが努力すれば守れること	・納得していないメンバーに対して、どの部分が納得できないのかをヒアリングし、問題点を明らかにした上で問題認識を全メンバーで共有する ・「しらけ」を抱えたままの場合、チーム全体として著しくモチベーションが下がってしまう場合がある

No.	知識エリア	チェック項目	評価基準	マネジメントにおけるヒント	対策案
S38	モチベーション	メンバーに疲弊感が漂っていないか？	プロジェクト・メンバーの表情、言動や後姿に疲れが見られないこと	例えば、もうろうしてと仕事をしていたり、作業にミスが増えたり、疲れた様子がある場合は、適切な対応策を実施する必要がある	・労務状況や抱えている作業量などを把握した上で、必ず週に1日は休養日を設けるようにしたり、作業の割り振りを見直して他メンバーへの移管を実施する。これは他メンバーの育成やチーム一体感の醸成に役立つ
S39	モチベーション	プロジェクト・マネージャ自身やメンバーのモチベーションの低下が見られないか？	プロジェクト・メンバーの表情、言動が暗くなっていないこと	問題がある場合、メンバーの仕事ぶりやメンバー間のコミュニケーションの様子などモチベーション低下の原因を探る必要がある	・モチベーション低下の原因をヒアリングする ・特に、問題のあるチームほどモチベーションが低下する傾向がある。そのようなチームに特別な支援体制を敷き、成功体験をさせる ・これは、他のチームのモチベーションを上げる効果もある
S40	課題管理	課題管理表などで課題が管理されているか？	課題（解決しなければならない既出の問題）の発生日時や課題の内容、重要度、状況、対策予定日が分かるような管理表やそれに類する資料によって課題を随時管理していること	課題管理表の現物を見ることが必要。 ・記載されている内容は本当に課題か？ ・課題の責任者が明確か？ ・解決までの経過が記述されているか？（目標日変更も含めて） ・解決目標日と完了日に矛盾はないか？ ・目標日を過ぎている未解決項目はないか？ ・特定の課題に偏っていないか？	・顧客側、開発側のキーパーソンを集めて課題を抽出し、一覧表にまとめる

付録 2. チェックシート【ヒアリングシート】

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H1	統合	計画が承認され、宣言されているか？	計画を説明するにあたって、その計画が顧客や関連部門、上位部門との間できちんと承認されているかどうかを明確にしていること。例えば、以下のような観点で確認を行なう。 ・キックオフ会議が開催されていること ・キックオフ会議に参加していたメンバーには開発者メンバーや上層部が入っていること ・途中から参加したメンバーにはプロジェクト計画について説明をきちんと実施していること	基本的に下流工程で対策を打つべきことではないが、それでもやる意味はある	・キックオフの議事録 ・説明記録 ・実質的なプロジェクト計画・テスト計画・移行計画は何かを確認	・プロジェクト計画・テスト計画・移行計画をステークホルダー間で確認し、承認してもらう
H2	統合	重要度の高いミッションクリティカルな機能に対して、適切なテスト計画が策定されているか？	テスト計画書にミッションクリティカルな機能のテスト計画が明記されていること	・ミッションクリティカル性の高い機能要件は失敗してはならない機能なので、計画・設計時から入念に管理され、実行され、監視される必要がある ・特にテスト内容が不十分の場合、実際の引渡し時や運用開始時に問題が発見された場合は大きな問題となるので、入念なテスト計画が必要である ・ミッションクリティカルなものは通常のガントチャートとは別にし、進捗も別に管理する ・俯瞰図のような進捗管理が必要である ・通常の機能でも、仕様変更によりミッションクリティカルな機能に変わる場合もある	・プロジェクト計画書 ・テスト計画書 ・重要機能リスト	・計画を早急に立てる ・特に、テスト計画は業務知識の豊富な高いスキルを持ったエンジニアに担当してもらう ・総合テストのレベルでは、運用にかかわるようなテストになるため、テスト内容については顧客側にも協力を要請することが望ましい
H3	統合	基本設計（外部設計、内部設計）の定量的評価結果（金額と納期を確定できたか）は問題ないか？	例えば、以下のような観点で確認を行なっていること ・当初の予算とスケジュールと想定規模は整合性が取れていること ・当初の予定と現状の差異に大きな乖離がないこと	・該当業務や技術に知見のある有識者でレビューすることが重要 ・問題がある場合、工程が遅延している可能性がある ・見積もりが正しくない場合にテスト段階で予算オーバー、工程遅延が発生する ・規模が測られていない場合、テスト量が予測できなくなる ・一番精度よく見積もれるタイミングで、予定と現状の乖離を分析したか？ その結果をもって、顧客と交渉できたか？ ・リソースの手配を考えたか？ ・顧客と交渉してだめだったときに何か算段をとったか？ ・顧客に押し込まれたとき、そのリスクに対してどういう処置をとったか？	・レビュー記録 ・リスク管理表	・基本設計で定量的評価がされていない場合は妥当性を確認しておく ・定量的に評価した結果、金額と納期に問題がある場合は、費用、機能、納期について、社内や顧客と調整する

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H4	統合	環境（作業場所、支援システム、ツール）についての見通し、または計画がきちんとしているか？テスト環境利用計画があるか？	プロジェクト計画書に明文化されていること	・一般には計画段階でこのようなことを検討して見通しをつけ、それに基づいて手を打っておくことが普通である ・環境準備の段取りが WBS になっているか？ ・パッケージ利用やオープン化のソフトウェアの組み合わせなど、設計段階で見えてなかったものが統合段階で突然出てくることがある ・本番稼働環境でしか確認できない事項との差異を見極めているか？ ・例えば、OS がサポートしてないミドルウェアを選定しているようなことがないか？ ・方式面で初物があるか？ 初物（未経験技術や今までにないソフトウェアの組合せ）やパッケージ利用には要注意。パッケージはカタログ・スペックと実際のスペックが異なる場合がある	・プロジェクト計画書 ・テスト計画書 ・テスト環境利用計画 ・これに関連する WBS	・今後の作業計画と照らし合わせて、作業場所、支援システム、ツールなどの環境面で不都合がないかを確認し、計画を立てる
H5	統合	プロジェクトの日程、責任範囲、WBSなどがステークホルダー間で合意されているか？	ステークホルダーが納得の上で合意していること。 例えば、以下のような観点で確認を行なう。 ・アプリケーション開発上の機能 / モジュール・レベルのリストと担当者が明確になっていること ・開発以外の作業項目（例えばマシンの手配、作業場所の確保、文書管理、変更管理など）について担当者が明確になっていること ・マスターデータの移行、テストデータの移行のメドが立っていること ・テストデータの準備は顧客の担当範囲かどうか合意していること	・プロジェクトの WBS と顧客の WBS を比較することが重要である ・顧客とプロジェクトとの協働作業でやらないといけない ・お互いの作業分担を明確にすることがこの項目の主旨である ・この時点で分担が明確になっていない作業はベンダー側の責任となる	・プロジェクト計画書 ・責任分担表 ・WBS	・あとで「聞いていない」ということにならないように、関係者を集めて日程と責任範囲について合意を得る会議を開催する
H6	統合	構成管理の対象（ドキュメント、ソースコード）と、それを管理する開始日付が明確になっているか？	成果物の管理がきちんとなされていること。 例えば、以下のような観点で確認を行なう。 ・ドキュメントが管理されていること ・ソースコードが管理されていること ・第1版がどの時期に承認を得たかなどをきちんと管理していること ・プログラム設計レベルの修正は、どの時点の仕様書まで遡って更新されるかが決まっていること	・マルチベンダーでやっている場合に1社が遅れると、順当に進まない場合がある。 - 統合できない場合はテスト計画の変更を行う必要がある - ベンダーが虚偽の報告をしている場合、構成管理そのものが正しく実施されていない - 要件が増加すると構成管理にしわ寄せがくる ・不良が出たときにどうするかなど構成管理のルールが必要である。 - バグ票が回ったときに追跡できるようにしておく - 問題点の追跡管理ができるか？ - 課題管理は優先順位を付けて行なうこと	・構成管理計画書 ・構成管理ツール ・運用ルール	・構成対象が不明確、あるいは管理されていない場合は、必要な構成管理対象をリストアップする。それらの最新のものを集め、ベースとすることをオーソライズし、管理を始める

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H7	統合	構成管理責任者が明確になっているか？ また、実際に構成管理を行っているか？	構成管理責任者が構成管理を実際に行なっていること	・構成管理責任者という肩書きよりも、構成管理について責任をもって把握している者がいるかどうかが重要である ・定期的に監査が行なわれているとよい ・「～してますよね？」ではなく、「～はどうやってますか？」というオープンクエスチョンで聞くこと ・構成管理をする際にステアリングコミッティや顧客を巻き込んだ運営管理が存在しなければ作る	・プロジェクト計画書 ・体制図 ・活動記録	・構成管理責任者を立てる
H8	統合	構成管理のルールや手順が明確にされているか？	構成管理のルールや手順が文書化されていること	特にバージョンが輻輳している場合、リリース・モジュールの管理などが手順化されていない場合は、リリースミスやリグレッションなどの問題が発生し、大きな手戻りになることがある	・ルール・手順書 ・リリース管理フロー	・リリース管理の徹底を行う ・リリースミスやリグレッションなどがひどい状態の場合には、まず実施手順を見直し、ミス低減を進めるとともに構成管理チームを別途作り、システム環境やソースを徹底管理する
H9	統合	保守・運用計画および作業手順・ルールを明確にし、レビューを実施しているか？	体制作り、保守・運用計画に問題がないかをチェックしていること	・プロジェクトのクロージングで問題になるケースがある ・統合テストより前の段階で確定しているか？ ・確定していない場合、次の項目を確認する - これから手を打てるようになっているか？ - 枝葉の部分で何とかできるものか？ - 建て直し計画の中に保守までやる必要はあるかないか？ ・日次、週次、月次など運用サイクルが考慮されているか？ ・運用訓練が計画されているか？	・保守・運用計画書 ・レビュー記録	・保守・運用に関する体制の計画を立て、関係者と合意しておく。 ・作業手順・ルールがない場合は過去の事例などを参考に整備する
H10	統合	本番移行計画が明確か？ （移行計画には、人的移行、システム移行、データ移行などがある）	顧客との責任分担を明確にしていること。移行計画や移行に関する作業についても顧客との合意を得ておくこと	ファイルの移行はもちろんのこと、顧客データを移行するときには、顧客データの品質に問題があることが多々ある。 顧客の保持しているデータの信憑性をよく確認して移行計画を策定することが重要である	・プロジェクト計画書 ・移行計画書	・本番移行計画を立て、有識者や顧客も含めたステークホルダー間でレビューを実施する

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H11	スコープ	システムの方向性を変えるような機能範囲と作業範囲の変更に対してきちんと管理しているか？	例えば、以下のような観点で確認を行なっていること ・変更が発生した場合の手順が明確になっていること ・窓口、取りまとめ者が明確になっていること ・実際の実施に関して問題がないこと	・あらかじめステークホルダーに変更管理手続きを徹底しておく ・しきい値やクライテリアがどうなっているかを確認する ・「2-4-3 の法則」はスコープの大局観を示している ・このチェック項目はプロジェクト・マネージャの緻密さを見ている ・プロジェクトでは、変更が発生することはよくあるが、成り行きで認めるのではなく、納期、コストとの関係からコントロールしていく必要がある ・システムの方向性が変わるかどうかがポイントである ・スコープが決まって WBS がスケジュールと工期に影響する ・システムのスコープが成り行きで変わらないようにすることが重要である	・変更管理表 ・変更管理ルール	・変更管理ルールがない場合は、変更管理についての手続きを明確にする ・変更管理の実施に問題があるかを関係者にヒアリングし、問題があれば改善する
H12	スコープ	想定外のスコープ増（機能範囲と作業範囲の増加）が発生しているか？ 発生している場合、それは許容範囲内か？	スコープ増を定量的に押さえていること。 許容範囲かどうかをコスト、スケジュールの観点から評価していること	・スコープ増加時の対応策（機能削減、費用増、スコープ変更）を顧客とあらかじめ合意しておく ・スコープ増の場合に、テストの増大やクリティカルな機能に対する影響度やどの時点でスコープ増が発生したかを確認すること	・変更管理表 ・議事録	・許容範囲でない場合、どの程度の増加があるのかを明確にした上で、顧客との打ち合わせを実施する
H13	スコープ	前工程担当者から引き継ぐときの理解度は十分か？	何が引き継ぎ資料なのか定義されていること。検討が十分でないところが明確にされていること	・引き継ぎ資料が不足なくあることや、その内容についてもきちんと理解する ・前工程のアウトプットを十分にそしゃくする ・もし理解度が不足している場合は、引き継ぎ資料のレビューや上流工程担当者からの再説明を行うなどの対策が必要である ・コンサルティング会社の成果物が SI ベンダーに渡った段階で引き継ぎされてないことがあり、プロジェクトの失敗がそこに起因する場合がある ・テスト結果を見て良いと判断できる人が何人いるかを V 字モデルに対応させて確認すること	・引き継ぎ資料	・理解度に不足がある場合は、コンサルティング会社に再度支援を依頼する

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H14	スコープ	他社（顧客を含む）が開発したシステムとの接続がある場合の責任分担は明確か？	他社が開発したシステムとの接続に関する責任分担表とテスト・移行スケジュールを関係会社と合意しておくこと。 例えば、以下のような観点で確認を行なう。 ・接続テストの日程について他社と合意が取れていること ・テストの作業順序について合意がとれていること ・テストデータはどちらが準備するか合意がとれていること ・テスト結果の取りまとめはどちらが行うか合意がとれていること	・自社の WBS か他社の WBS か、担当分担は明確になっているか？ ・バグ票のフローのルールなどが明確か？ ・誰がテストの責任者かを明確にする ・5W 1H を明確にする ・テストの目的、テストを行なうまでの段取りをはっきりさせる	・プロジェクト計画書 ・体制図 ・責任分担表 ・テスト・移行スケジュール	・責任分担と接続に関するスケジュールを明確にし、顧客、他社、自社の３者での打ち合わせを実施する
H15	スコープ	仕様変更対応の（契約上の）取り扱いについて顧客と合意し、別精算などの対応を行っているか？	顧客と文書で合意がとれていること	・例えば、承認ルート、精算方法、リスク・インパクトの合意などを考慮していること ・契約の問題。業界常識を後工程では言わない（流通業、金融業で多い） ・この項目の趣旨は契約がきちんとしてないとトラブルが発生するということ ・スコープの問題を契約で明記しておくこと。１次請負会社もそうだが,２次請負会社も同様。２次請負会社以降がきちんとしないと、１次請負会社もきちんと対応できない	・議事録 ・承認ルート ・精算方法	・契約上の取り扱いについて合意が取れていない場合、開発を一度止めて、早急に合意をとる
H16	タイム	実施するべき作業が明確に定義されているか？	いつまでに、何を行なうかが明確になっていること	作業のリストアップができていないか、あるいはリストアップはできているが、その作業で何をするのかが明確にされていないか？	・スケジュール表 ・WBS ・見積もり ・責任分担表（体制表）	・キーパーソンを集めて、やるべき作業のリストアップを行う ・リストアップされた作業について、作業内容と成果物を明確にし、作業量の見積もりと作業スケジュールを明確にする ・担当者をアサインする

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H17	タイム	担当者の作業負荷や作業の重なり具合、空き具合についてチェックしているか？	プロジェクト・マネージャと担当者の双方がチェックしていること	・チェックしていないと管理者が担当者ごとの作業負荷に関し無頓着になり、計画と整合がとれていないことがある ・進捗を何で見ているかが重要である ・きちんと見ているか？ 資料を一生懸命作っていても、誰も見ていないということがないか？ ・現物（実物）を見ているか？ ・モノがないのに終わりになっていないか？ ・終了したことを何で証明するか？ エビデンス、終了基準（工程完了基準）が規定されていることを実物で確認すること ・90％症候群（いつまで経っても100％にならない）にならないこと	・スケジュール表（進捗管理） ・WBS ・見積もり ・責任分担表	・進捗会議などで作業負荷状況を定期的にチェックするようにする
H18	タイム	工程進捗のつじつまは合っているか？	例えば、以下のような観点で確認を行なっていること ・各作業項目のスケジュール表（小工程・ガントチャートなど）で前後関係に矛盾がないこと ・基本仕様書も総合テスト計画もできていないのに総合テストが開始されているような場合がないこと	・クリティカルパスの部分とそうでない部分が明確になっているか？ ・初期の時点だけでなく、変更が発生したときにそれに対応した変更ができているか？ ・修正が局所的になっていないか？	・スケジュール表 ・WBS ・見積もり ・責任分担表	・全体線表と各作業項目の進捗具合をチェックし、つじつまが合っていないようであれば、作業を一度止めて、見直しを行い、先にやるべきことを実施する
H19	タイム	マスタースケジュールと要員山積みとの整合性は十分か？	マスタースケジュール作成時に要員山積みを作成して整合性を確認していること	当初の計画との差異を見ていること。開発のV字構造に照らし合わせてテスト要員が山積みの中に入っていること。具体的には、詳細設計の要員が結合テストを担当していること	・マスタースケジュール ・要員山積み表	・要員調整計画を見直す
H20	タイム	マスタースケジュールとチーム別スケジュールとの整合性は十分か？	マスタースケジュール作成→チーム別スケジュール作成→マスタースケジュール改訂のステップを実施して、両者の整合性がとれていること	・複数のベンダーが出したスケジュールの一つに変更あった場合にコントロールできているか？ スケジュールを統合したときに、おかしくなっていないか？ ・各社のスケジュールが変更されたときのマスタースケジュールへの影響はどうか？ マスタースケジュールとの突き合わせをしていないと丸投げである ・この項目は、丸投げしていないかをチェックする項目である ・作文としてのスケジュールは意味がない。スケジュールに根拠があるかどうかが重要である ・2人で3カ月かかるものが3人で2カ月で終わるとは限らない	・マスタースケジュール ・チーム別スケジュール	・マスタースケジュール作成→チーム別スケジュール作成→マスタースケジュール改訂、というステップを実施して両者の整合性をとる

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H21	タイム	作成したスケジュールに根拠があるか？	作業量と体制との関連づけが明確になっていること	・根拠をどこまで考えているか？ ・プロジェクトに対するプレッシャー度合にもよる。顧客のプレッシャー、SI ベンダーのプレッシャー、上司のプレッシャーが重なって出てくる ・2 次請負会社以降は、仕事を請け負う際に、その工程が妥当かどうかをどう判断したか？ ・能力とスキルを考えているか？ ・SI ベンダーが何を根拠に発注しているか？ ・1 次請負会社は 2 次請負会社に根拠をただしているか？	・スケジュール表を書いた根拠	・なぜ根拠のないスケジュールを作らざるを得なかったかという背景を探る - 顧客、1 次請負会社、上司からの根拠のないプレッシャー - プロジェクト・マネージャの知識 / 経験不足 - メンバーの言いなり - プロジェクト・マネージャの意思なし ・無理矢理引いた(引かされた) だけのスケジュールは再スケジューリングしかない 【再スケジューリングのポイント】 ・作業要員、個人別作業能力と、集中度（過負荷にならない）を勘案した再スケジューリングを行う ・予算との見合いで要員増強・期間延長を考慮する ・リスクに耐えられるスケジュールにする ・必ず途中チェック、レビュー、修正期間を入れる ・初めに納期ありきの場合は、リスクとなる可能性が非常に高い。フェーズごとに、それ以上は短縮できない最低作業期間がある。その期間をクリアするようにする
H22	タイム	線表の稲妻線の根拠は明確か？	進捗率の値が感覚的なものではなく、きちんとした根拠に基づいたものであること	・前後関係の論理矛盾の確認に対してつじつまが合っていることを見る。稲妻線は予実績管理を見る ・テストケース数の粒度は問題ないか？ ・シナリオや業務ケースを作成しているか？ ・進捗の根拠と計画は問題ないか？	・テストケース消化率 ・バグ票 ・スケジュール表 ・WBS	・進捗の評価方法を形式化 / 文書化し、進捗報告者に徹底させる

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H23	タイム	クリティカルパスは明確か？	クリティカルパスに当る作業項目が洗い出せていること	・クリティカルパスの遅れはプロジェクトに大きな問題を引き起こしやすい ・クリティカルパスをどうやって見るか。PERT 図がいつも正しければ分かるが、各作業で実際にかかる時間には幅があるため、出たとこ勝負の面もある ・計画時には PERT 図を作るが、維持している例は少ない ・クリティカルパスになりそうな部位を特定できているか？ - プロジェクトにとって足をひっぱられそうな個所はどこか？ - 過去のクリティカルパスがどうなっているか？ - テストの中のクリティカルパスを管理しているか？ ・クリティカルパスは毎日変わるから難しい ・ボトルネックは何か、という視点で見た方がよいことがある。ボトルネックの例として、誰かに偏ってないか、遅れが偏ってないか、品質が偏ってないか、がある。一人で背負い込む人がボトルネックになる ・危ない個所にはスキルを持った要員をつけること。キーパーソンと思っていたのに違ったという失敗事例につながる ・危ない分野が特定できているか。危ない個所に変化があるか	・スケジュール表 ・WBS	・クリティカルパスに遅れがある場合はキーパーソンを集め、早急に問題点の洗い出しを行い、対策を取る ・クラッシング、ファーストトラッキングなどの手法がある
H24	コスト	コスト制約については、どこまで調整が可能か？	コスト制約に見合った機能削減や請負金額の増額、スコープ変更などの対応実施策を顧客と決めておくこと	・例えば、請負金額はどこまで増額が可能なのか、機能削減をするか？ ・基本契約ごとに付く個別契約の特記事項として記述されているか？ ・基本契約は会社ごとに決定している場合が多い。きちんとできていないとコスト、スケジュールに影響してくる。なかなかきちんとは書けないが、最近は発注者側も認識してきている	・プロジェクト計画書の品質目標に関する項目を確認する ・記述がない場合はそのこと自体のリスクが高い ・契約書の附表	・コスト制約に見合った機能削減を実施することを顧客と決める
H25	コスト	複数視点での見積もりを実施しているか？	複数の見積もり手法を使ったチェックや経験十分な複数人による見積もりチェックによって、妥当性を確認していること	複数の見積もりを実施することによって、単一の見積もりでは気付かなかった重要な要件、実現可能性、考慮すべきリスクなどに気が付くことがある	・見積もり書	・見積もり方法が単一の場合や経験に基づくものである場合、現実の規模や工数が想定からかけ離れている可能性がある ・当初予想していた規模と現状の規模を見比べて、大きく離れているようであれば、プロジェクトの今後の計画を見直す必要がある

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H26	品質	コンポーネント→サブプログラム→システムと段階的にテストを行うための具体的な方法が計画されており、実施されているか？	ハードの搬入よりも先に、結合テストを実施するなど、矛盾のあるスケジュールになっていないかを確認していること	ある一つのコンポーネントのテスト遅延がシステム全体のテスト遅延になる場合もあるので、事前に全体のテスト計画を立てていることが重要である。 テスト計画の具体性も含めて確認し、問題がないことを確認する	・全体テスト計画 ・テスト計画書 ・テスト実施手順書	・テスト計画に問題点がある場合は、問題点およびリスクの洗い出しを行い、解決方法およびリスク回避方法を検討する
H27	品質	テストケース、テストデータ、テストツールの妥当性を確認しているか？	例えば、以下のような観点で確認を行なっていること ・ケースやデータに偏りがないこと ・テストの網羅性に問題がないこと ・正常系のみのテストケースになっていないこと ・テストデータの内容はテスト内容に対して妥当であること	・テストケース抽出基準とそのレビューの実施を確認する ・テストツールのカバー範囲を認識しているか確認する ・シナリオによるテストを計画・実施しているか？ ・シナリオは妥当か？ 特に、特異な運用パターンを忘れないようにする	・テストケース表 ・テストデータ ・テストツール ・テストレビュー記録	・偏りがある場合には是正する。 ・追加テストの実施、ツールの用意をする
H28	品質	コードインスペクション（コードレビュー）の結果は、既定の方法で管理され、品質や生産性の目標が満たされているかどうか監視しているか？	例えば、以下のような観点で確認を行なっていること ・コードインスペクションでの問題摘出件数の指標などが決まっていること ・コードインスペクションではどのような視点で問題摘出するかが明確になっていること	コードインスペクション結果に問題がないことを確認できるようにしておく。ピアレビューなども有効である	・レビュー記録	・コードインスペクションの結果が管理されていない場合は、管理するための帳票などを準備して記録・管理し、指摘件数などを指標に品質評価をする
H29	品質	ロジックの網羅性や、データ、日付などの境界を意識したテストをしているか？	例えば、テスト項目として以下が含まれていること ・網羅率の測定 ・境界値を用いたテスト ・うるう年のテスト	ツールの利用状況などを確認する	・テスト結果報告書	・カバレッジ計測ツールなどを利用して網羅率を測定する ・単体テストが人手で行われている場合は、カバレッジ計測ツールとともに、自動化するための仕組みを導入する
H30	品質	各テスト工程に対するテスト計画が明確であり、計画通りに作業を行っているか？	無計画なテスト作業を行っていないこと。 例えば、以下のような観点で確認を行なう ・全体的なテスト計画を基に、テストを実施していること ・テスト結果のレビューについて計画通りに実施していること ・テストで手戻りが頻発していないこと（リリースミスや実装ミスにより同じテストを何度も実施していないこと）	・テスト計画を策定する際にシステム規模が当初と変わっていないか確認する。そうしないとテスト規模の見積もり誤りを起こす ・テストの戦術を明確にすること。重箱の隅を先につついても仕方ない。顧客が常に使うところから先に品質を落ち着かせるべきである ・ライブラリを入れ替えたときには再度テストを行なうかどうかチェックする必要がある。これを誤ると、思わぬデグレードの事故が起こる可能性がある	・テスト結果報告書 ・プロジェクト計画書 ・テスト計画書	・テスト計画が明確でない場合は、テスト計画を立て直す ・テスト計画は明確であるが計画通りに作業していない場合は、その原因を特定し、要員に問題がある場合は指導を徹底する ・外部的な要因（他社・顧客からの依頼で優先度を変えたなど）の場合は、予定変更の報告を義務付ける ・外部要因に対して計画変更の理由を確認する ・必要に応じて今後のテスト計画を見直す

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H31	品質	バグ票により不良が管理されているか？	バグ票の管理方法をプロジェクト・メンバーに徹底していること	バグ票の記入方法や記入内容が妥当かどうかをチェックし、必要であれば担当者に入力方法を徹底する	・バグ票	・バグ票管理がされていない場合は、バグ票の利用と手順の順守を担当者に徹底させる
H32	品質	事前にテスト検証作業の手順、方法を計画した上で、テスト検証作業を実施しているか？	テスト検証作業の手順、方法がプロジェクト・メンバーに周知・徹底していること	プログラムのコードレビューやテスト結果に対する検証作業が行き当たりばったりではなく、事前に計画されており、検証作業もそれに従っていることが重要である	・手順書 ・テスト結果報告書	・手順、方法に従っていない場合は、順守するよう徹底させる
H33	品質	テスト（結合/総合）計画とそのレビュー結果は問題ないか？	レビューで出た指摘事項に対して対処を行なっていること	品質評価指標と目標値を事前に定めた上で、テストケースをレビューする	・プロジェクト計画書 ・テスト計画書	・テスト計画のレビュー結果に問題がある場合は、不足しているテストケースをリストアップする ・もし専門知識（業務知識やシステム基盤など）が必要なテストケースが洗い出せていない場合は、有識者にテストケース作成の協力を依頼する
H34	品質	本当にテストされているかを確認するために、テスト項目の消化実績を検証しているか？	消化日や実施結果など、テスト実績を残していることを確認していること	テスト結果の確認方法が明確になっている必要がある。 テスト結果が NG だった場合にはバグ票が発行され、テスト項目とバグ票との対応が明確になっている必要がある	・テスト結果報告書	・テスト実績を残していない場合は、今後のテストでは実績を残すようにする ・クリティカルパスについてテスト実績が残っていない場合は、再テストをするなどの調整を行って、品質確保を徹底する
H35	品質	本番環境でテストを実施する計画が策定されており、本番環境での確認項目が作成されているか？	本番環境でなければテストできない項目を明確にしていること	本番環境でテストを実施できない場合は代替案を検討する必要がある	・プロジェクト計画書 ・テスト計画書	・本番環境でのテスト計画が未計画の場合は、計画を立てる ・本番環境でテストを実施できない場合は代替案を検討する
H36	品質	性能要件を満たしていることを確認するためのテスト計画が明確か？	運用テストや実際の運用に入ってから性能問題が発生しないように、事前に性能テストを実施していること	本番環境でないと実施できない性能テストの場合もあるが、性能テストを計画することで、課題を認識したり、課題を解決するためオーバーフローの対処策を検討し、実装したりすることができる	・プロジェクト計画書 ・テスト計画書	・性能テストの計画が未計画の場合は、計画を立てる ・性能目標値が明確でない場合は、顧客との打ち合わせを実施して明確にし、合意する
H37	品質	障害対策、運用面でのテストの計画が明確か？	実際の運用に入ってから障害問題が発生したときの対応項目を事前にきちんと挙げていること	実運用面での操作・作業ができるかどうかを事前に確認する	・テスト計画書 ・運用設計書 ・通常運用書 ・障害運用書	・障害対策、運用テストの計画が未計画の場合は、計画を立てる

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H38	品質	ドキュメント作成やコーディング時に従うべき基準が明らかにされており、その基準に従って作業を行っているか？ また、プロジェクト・リーダーは開発者に基準の順守を、随時促しているか？	社内に標準的な基準・フレームワークがある場合はそれに従っていること	ステークホルダー全員が基準の存在と利用方法についてきちんと理解し、利用していることが重要	・コードレビュー結果 ・ドキュメントのひな型と実体の比較 ・ソースコードとコーディング規約の比較	・コーディング基準書の重要性をリーダー自身に理解させ、徹底を促してもらうよう依頼する ・基準となるベースがない場合は過去のプロジェクトで利用したものを活用するなどして、標準化する
H39	品質	計画時に規定されたプログラム変更票の作成工程に従い、プログラム変更票を作成、レビューを行っているか？	勝手な思い込みや独自の判断でプログラムの修正を行っていないこと	プログラム変更は、なぜ変更する必要があるのか、変更による他の部位への影響度合いなどを明確にする必要があるので、プログラム修正はプログラム変更票によって管理される必要がある	・プログラム変更票	・プログラム変更票の活用を励行する ・また、ソース改変時にプログラム変更票との突き合せを実施して、きちんとプログラム変更票が作られているかを確認する
H40	人的資源	各作業に担当が明確に割り当てられているか？	誰が、何を行なうかについてプロジェクト・マネージャと各担当者が認識を共有していること	・誰がその作業をするのかはっきりさせておかず、土壇場になって誰にやらせるかということで問題になる ・WBSの作業には、自社でできるもの、他社との関係で作業するものがある ・失敗プロジェクトは協働で作業している場合が多い ・責任分担は明確でないといけない	・体制図 ・スケジュール表 ・WBS ・責任分担表	・特定のメンバーに負荷が集中している一方で、遊んでいるメンバーがいる場合は、作業の割り振りを見直したり、多忙なメンバーが抱えている作業を他者に回すために努力したりする
H41	人的資源	他部署と連係して作業を行う場合、作業範囲と役割分担、作業を推進する責任者がステークホルダー間で明確か？	例えば進捗報告や会議体などを推進する責任者が明確になっていること	自社でも他部署との関係がはっきりしていること	・進捗報告書	・取りまとめは誰が責任を持って行うのか、ステークホルダーのそれぞれの位置を明確にする
H42	人的資源	品質保証部門やプロジェクト外のステークホルダーからの指摘、意見に対して真摯に受け止めて対応（必要ならば是正）を行う責任者が明確か？	課題や問題に対する対応策が計画されていること。 または、議論による問題解決や課題・障害の解決に向けてのステークホルダーの説得ができていること	・隠ぺいしようとしている部署はたちが悪い ・会社によって品質保証部門の責任の持ち方が異なる。品質の責任を持つのはプロジェクト・マネージャであるが、プロジェクトが大規模でプロジェクト・マネージャが一人でできない場合は専任の品質保証担当者を置くこと ・PMBOKでは、品質管理は経営者の責任である ・品質は利益であると考えることが重要である	・課題管理表	・是正処置が行われていない原因を特定し、指摘事項の重要度と影響度を考えて対応策を実施する ・例えば指摘や意見に対し、是正処置を実施したいが負荷が高すぎて実施できない場合もあれば、スキル・要員が不足していることが原因の可能性もある ・指摘事項の重要度を考えて、改善策を検討、実施する

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H43	コミュニケーション	顧客、顧客側キーパーソンなどの要求仕様を十分に考慮し、必要に応じて要求仕様のレビューが行われているか？	顧客側の要求仕様に対する理解に問題がないかをチェックしていること	特に顧客からの要求仕様に対しては打ち合わせを持つなどの直接的な会話で内容を確認するべきであり、文書のみの依頼で要求仕様を理解した気になってはいけない	・レビュー記録 ・議事録 ・要求管理表	・顧客との要求内容に関する打ち合わせを行う
H44	コミュニケーション	作業内容と終了基準についてのメンバーの認識があいまいではないか？	例えば、以下のような観点で確認を行なっていること ・単体テストはどこまでやったら終了か、明確にしていること ・無計画なテストをしていないこと	あいまいさがあると作業の進捗が把握できなかったり、把握したと思っていてもメンバーにより認識の内容が異なっていたりして、プロジェクトとしての統一した進捗把握ができない。品質にも問題が出ることがある	・作業内容の終了基準を確認	・認識があいまいになっている場合は、どのような状態になれば終わりを宣言してよいかのイメージを明確にし、作業者に伝える ・また、作業進捗状況を定量的に測定するための方法を考え、定量的な進捗報告を徹底する
H45	コミュニケーション	開発チーム間・協力会社とのコミュニケーションに問題はないか？	適切なコミュニケーション計画を立案し、これに従っていること	会議体が定義され、実施されていることを確認する	・会議がどれだけの頻度で行われているかを議事録などから確認する ・協力会社へのヒアリング	・コミュニケーションが円滑でないと判断した場合、適切な会議体を設置する ・担当者間の相性に問題がある場合、1対1でのコミュニケーションには問題が出やすいので、第三者（プロジェクト・マネージャなど）を交えた会議体を定例化する ・3次請負会社（孫請け会社）が、発注元である2次請負会社を介してしか1次請負会社と話をしないために、コミュニケーションが滞りがちになる場合は、階層構造をフラットにし、各社からのキーパーソンを集めた会議体を作る
H46	コミュニケーション	顧客とのレビュー議事録などがあり、きちんと合意（承認）を得ているか？	顧客との各種レビューを実施し、議事録を作成し、その議事録について合意を得ていること	・顧客とのレビュー計画書は作成しているか？ ・レビュー実施要領は作成できているか？ ・実施要領などの中で確認方法を取り決めているか？ ・議事録またはメールでも記録として残すこと	・議事録	・レビュー議事録を作っていない場合は作る ・報告会議などで議事録のレビューを慣例化する
H47	コミュニケーション	顧客などのステークホルダーと共有すべきプロジェクトリスクについて、共有を行っているか？	リスク管理表をステークホルダー間でレビューすること	顧客と作業の責任分担を明確にしたうえで、期日が守れなかったときの対策協議を明確にしておく必要がある。そうしておかないと、水かけ論に終始して、すべてを自分でやらなければならなくなる可能性がある	・議事録 ・リスク管理表	・リスクに対する意識を持たせるために、ステークホルダーとの打ち合わせでリスク管理表をレビューする

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H48	コミュニケーション	重要な指摘が発生した場合に、指摘事項が全員に周知徹底されて、再発防止を図っているか？	例えば、コードインスペクション結果、障害情報などの情報共有の仕組み・やり方が決められていること	同じような間違いやミスが他の部分にもある可能性があるので、情報共有を行っていることが大切である	・議事録 ・レビュー記録 ・周知徹底の記録（メールなど）	・コードインスペクション結果、障害情報など重要な指摘事項を周知徹底させることを義務付ける
H49	コミュニケーション	性能テストの結果について、顧客の合意が得られているか？	レビュー結果は議事録に記載して承認してもらっていること	性能テストの結果について、顧客へ報告し、問題がないかを判断してもらっておく必要がある	・議事録	・合意が得られていない場合は、妥協点の調整をする一方で、早急に性能改善を検討する ・特にクリティカルパスに関連する性能問題については定期的な監視を行って、改善活動をコントロールする
H50	コミュニケーション	マイルストーンについて、顧客との間でコンセンサスがきちんと得られているか？	マイルストーンの内容は議事録に記載して承認してもらっていること	マイルストーンやイベントなどについては、議事録に残すだけではなく、線表にまとめ、顧客側関係者と開発側関係者で共有して意思疎通を図る	・議事録 ・レビュー記録 ・周知徹底の記録（メールなど）	・マイルストーン設定内容について顧客から合意を取る
H51	コミュニケーション	スケジュール・作業内容について顧客との認識（作業レベルの意識）にズレがないか？	合意を得ていること	顧客に対しスケジュール・作業内容を説明するためのレビュースケジュールを確保していること	・議事録	・スケジュールの妥当性についてまずは内部で確認・合意し、その後顧客とともにレビューする ・問題がある場合はスケジュールを見直す
H52	コミュニケーション	開発者とテスト担当者の間で障害情報、修正情報が共有されているか？	障害管理ツールなどで、修正情報がテスト担当者にも閲覧できるようになっていること。テスト担当者が最新の仕様書の所在を知っていること	情報が共有されていない場合、障害が修正されていないにもかかわらず、再テストしてしまう、修正したにもかかわらずテストされないなどの作業のロスが発生する	・リリースアナウンス ・周知徹底の記録（メールなど）	・リリースアナウンスなどのルールを設定し、関係者に周知する
H53	コミュニケーション	顧客との「仕様」、「価格」、「納期」に関するやり取りは文書で行なっているか？	仕様、価格、納期に関するやり取りは文書以外で行なっていないこと	口頭による約束では「言った」、「言わない」の問題になる	・報告書 ・議事録など	顧客とのコミュニケーションで重要な項目は文書でやり取りする

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H54	リスク	プロジェクト規模はどのくらいか？	次のうちのどれに該当するか？ ・300 人月以上 ・100 ～ 300 人月 ・100 人月未満	・100 ～ 200 人月程度の粒度で疎結合（オンライン接続を極小化）するようなサブシステム分割ができないかを検討する ・プロジェクトのフェーズ分けができないかを検討する ・COCOMO、PUTNUM などの指数モデルによる見積もりを行う ・当初計画との規模の差異は出ていないか？ ・規模が拡大していないか？ ・機能を増やしていなくても、見積もりミスをしている場合もある ・差異に対する対応は取れているか？ ・間違っている場合は、どう修正していくか？ ・膨らんできた要件をどう削るか、顧客と調整しきれているか？ ・納期が間に合うような規模の中に納まっているか？	・サブシステム分割が分かる資料（基本設計書など） ・体制図	・計画段階ではフェーズ分けなどが可能かもしれないが、中盤以降では分けるのは困難 ・プロジェクトの規模が大きい場合は、必要に応じてプロジェクト・マネジメント・チームの強化策を実施する
H55	リスク	脅威を減少させるために、リスクの分析と影響度の優先順位付け、対策について十分検討しているか？	個々のリスクについて客観性のある分析を行った上で、対策を検討していること	・リスクは回避するものではなく、必要に応じて引き受けるものである ・顕在化したリスクがないか？ ・リスクが顕在化したならば、対応策がとられているか？ ・対応策とは人を増やす、納期を調整する、機能を落とす、など ・機能単位に納期を調整する余地がないか？ ・危機状態に顧客を巻き込めているか？	・リスク管理表	・リスク管理を行っていない場合は、リスク管理表を作成して、回避策を検討する
H56	リスク	リスクの内容について、その内容が明確に記述されているか？	リスク管理表にリスクの内容が記載されていること	・例えば顧客との打ち合わせに出席している者でないと意味が分からないような記述になっていないか？ ・本来、下流工程ではリスク・マネジメントは間に合わない ・リスクそのものの把握を行なうこと ・リスクのトリガーを明確にしているか？ ・対策を明確にしているか？ ・クライシスが発生してからドミナント・アイテムを探せるか？ ・大抵は、人の問題に帰着する	・リスク管理表	・リスク管理表上、あいまいな記述や誤解を生むような記述がないかをチェックする。そのような記述があれば、記入者を特定して、真意をヒアリングし、分かりやすく明確な記述にする。 ・その後、再度レビューを実施し、リスク内容をステークホルダー間で共有する
H57	調達	協力会社の進捗・品質に関する中間フォロー実施計画が明確になっているか？	進捗会議の定例化や品質基準の明確化についての計画がされており、ステークホルダ間で合意されていること	・請負契約は、成果物さえ納入すればよいと考えていないか？ ・成果物の品質が確保されないことが大きな問題という認識を持っているか？ ・協力会社の品質方針・品質体制・具体的な品質管理について確認が必要 ・中間時点（定期的）のチェックを約束･実施しているか？	・進捗管理表 ・委託契約書（提出すべき中間成果物などが記述されていることがある）	・協力会社へのフォローに関して問題がなければよいが、フォローを行っていない状態である場合は、フォローの計画を立てて、実施する

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H58	調達	実施責任者(プロジェクト・マネージャ、プロジェクト・リーダーなど)および有識者を交え、協力会社に依頼する作業が適切な規模、難易度であることを検討しているか?	協力会社の要員数、技術・スキルのレベル、これまでの実績、業務ノウハウの保有について、問題がないかを議論していること	・協力会社の力量を十分つかんでいるか? ・その上で仕方なしに発注したとしても、その状況を週1回の定例会議だけでなく、実態を押さえること ・報告をうのみにしてはいけない ・発注する際に約束事項を明確にしなければいけない ・例えば、不良が予定以上に出た場合、品質向上策を実施する、瑕疵責任をとらせるなど	・プロジェクト計画書や調達計画書など ・これらの計画書のレビュー記録など ・協力会社のマネージャとの面談結果	・もし十分な検討をしないまま発注をしているようであれば、行動を改める必要がある ・もし検討していないようであれば、すでに発注済の協力会社に関して、適切な規模・難易度かどうかを確認したほうがよい
H59	調達	協力会社への発注の根拠が明確となっているか?	業務知識、開発スキルなどの観点から、なぜその協力会社に委託することにしたのかを明確にしていること	例えば新規委託先の場合に、何を根拠に選定したのかを明確にする	・調達計画書 ・選定理由書	・発注の根拠を明確にして、リスク対策に反映させる
H60	調達	協力会社に対して追加発注の確約をしないまま追加作業を依頼していないか?	追加発注の確約をしないまま口頭で追加作業を依頼していないか確認していること。 契約がないのに、活動している協力会社の要員がいないか確認していること	確約がない場合は本当に作業を進めてよいか悩んだり、後の契約問題に発展したりする可能性があることから、作業着手を遅らせてしまうなどの問題が発生する	・発注書などの契約文書 ・WBS ・内示書	・追加発注を確約するか、作業内容の重要度などを個別に説明して対応を依頼する
H61	顧客	仮リリースの内容と仮リリースの条件が明確となっているか?	「仮リリース」とは、顧客の要望などにより本格リリース前に制約条件や前提条件のもとでリリースすることで、その内容と諸条件が明確になっていること	次の内容について顧客と合意をとっておく。「目的」、「前提条件・制約事項」、「機能」、「日程」、「正式仕様・正式プログラムとの関係」など	・議事録 ・仮リリース仕様書	・プロトタイプであることを顧客に明示する ・プロトタイプとは、何かを試すためのものなので、そのテスト目的と実際の提供物が乖離していると意味がない ・プロトタイプで提供できる機能一覧を作成し、それについて顧客側とレビューを実施して問題がないことを確認する
H62	顧客	顧客のプロジェクト推進能力は十分か?	顧客のプロジェクト推進能力とは、顧客の決断力、意思決定の早さのことで、それらが十分なレベルであること	現場のコンセンサスを得るための活動や意思決定のスピード、タスクの管理などが実行できているか?	・ヒアリング ・ステアリングコミッティ ・体制図	・顧客内のプロジェクト・コンセンサスを確認する場を設定するよう要請する ・また顧客タスクを文書化し、実行部隊のキーパーソンに徹底するよう要請する ・開発とは別体制の顧客プロジェクト・マネージャ支援契約を提案する
H63	顧客	顧客のキーパーソンを押さえているか?	例えば、以下の点を確認していること ・顧客内のシステム部門とユーザー部門との関係を把握していること ・仕様決定権限、予算、経営層へのコミットはどの部門が握っているかを把握していること ・決定権・発言力は誰が持っているのかを理解していること	リスク管理において、問題が発生した場合に協議すべき人を個人名で管理しておく。 責任部門のキーパーソンが必ずしも決定権を握っているとは限らないので、キーパーソン間の影響関係を把握しておく必要がある	・ヒアリング ・ステアリングコミッティ ・体制図	・顧客側の体制を確認し、プロジェクトに関連するキーパーソンには定例会議に出席してもらう

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H64	顧客	RFP の内容は十分か？	例えば開発範囲、契約条件、教育訓練、検収、スケジュールなどが明確に記載されていること	顧客企業が作成した RFP の場合、プロジェクトの定義、アウトプットの定義、要求獲得方法などが明確に記載されていること。コンサルティング会社などが作成した RFP の場合、顧客がその内容を十分理解していること	・RFP	・不足分については顧客側に要求する ・内容の不足分をリストアップし、顧客に申し入れを行う
H65	顧客	現行機能の保証が必要なとき、現行機能のドキュメント整備状況は十分であるか？	保証しなければならない現行機能については顧客から文書の形式で仕様書を受領していること	「現行機能を保証すること」という要望については、範囲があいまいになりがちである	・現行機能の仕様書	・現行機能のドキュメントが入手できない場合は保証ができない旨を伝え、合意する ・現行機能のドキュメントがあっても、新システムへの移行の問題で実現できない機能などがあれば、顧客と合意をとっておく ・現行機能の検証として、何をもって完了とするかの基準を顧客と合意しておく
H66	顧客	業務改革を含む場合、計画時に実現可能性について検討したか？	業務改革におけるシステムの位置づけと期待される役割を確認していること	業務改革の内容は実現可能性上問題がないことを検討し、その内容について顧客との合意がとれていることが重要である	・プロジェクト計画書 ・リスク管理表	・未検討の場合、プロジェクトの中盤以降であっても、業務改革についての実現可能性の検討を行って、問題がありそうならリスクとして管理する ・現場レベルの調整が必要になる可能性があるので、顧客と認識を合わせておく
H67	技術	市販製品、他プロダクトの活用について、すべての工程におけるシステム開発で、技術的観点から評価できる有識者を交えたレビューが実施されているか？	レビューで出た指摘事項に対して対処を行なっていること	パッケージ製品によっては品質が悪いことがあり、かつその中身がブラックボックスになっていて、容易に変更できない。このため製品の品質の悪さが性能問題や機能レベル不足につながる	・プロジェクト計画書	・レビューが未実施の場合、外部調達製品に問題がある可能性がある ・プロジェクト中盤以降であってもリスク分析という観点で再度見直す ・製品ベンダーとのチャネルを確保しておく ・製品の入れ替えを含む代替手段を検討する ・対応のコストを顧客と合意しておく
H68	技術	新技術 / 未経験技術があり、その技術に対する対応策は十分か？	机上評価、社内外の実績を評価し、開発着手前に実機評価をするようスケジュール化すること	・過小評価、過大評価に注意が必要である。冷静な判断をしているかを見る ・評価者自体が新技術を理解していない可能性があることに注意が必要である ・社内に評価・支援組織があるか？	・プロジェクト計画書	・新技術や未経験技術に対して、開発チーム側として対応に問題があるようであれば、有識者をアサインするなどの対策を打つ ・新技術・未経験技術であることをスケジュールに反映させておく

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H69	技術	移行切替方式（現新システム並存、他社からの移行など）については十分に検討しているか？	有識者のレビューを受けて、移行方式の実現性などをチェックしていること	可能な限り単純な移行切替方式を検討することが重要である。 顧客との役割分担を明確にし、契約形態を適切に使い分けることも考慮する	・プロジェクト計画書 ・リスク管理表 ・移行計画書	・移行方式が困難である場合、リスクを洗い出した上で顧客と打ち合わせを行い、段階的な移行を検討する ・不測の事態に陥ったときのリカバリ方式を検討して、移行失敗に対するリスクを低減する策を練る
H70	技術	レスポンスの確保や無応答状態をなくすなどの方式設計について十分に検討しているか？	例えば、大量アクセスがあった場合のスレッド枯渇や、障害発生時のクラスタによる二重化などの施策が考慮されていること	アプリケーションの機能設計ばかりに注目しすぎて、実際のレスポンス確保や無応答状態の回避など、非機能要件に近い要件に対してソフトウェア / ハードウェアの両方の観点からきちんと検討されているか？	・基本設計書	・方式検討がされていない場合は、方式に問題がないかをチェックする ・システム基盤技術の知識や運用に長けた外部ベンダー、コンサルタントに協力を要請するなどの対策を打つ
H71	技術	トラフィック量とトラフィックパターンについて適切に想定しているか？ どのような対応策をとるか明確になっているか？	クライアント数やアプリケーションのトラフィック量、ミドルウェアが利用するトラフィック量など、システム全体としての要求負荷や、ピークを想定していること	・顧客と話し、最悪のトラフィック量を検討しておく必要がある ・想定以上のデータが入ってきたときに閉塞を考慮する ・責任範囲を明確にする	・要件定義書 ・基本設計書	・顧客から現状の運用に関する情報を集めて、想定されるトラフィック量とパターンを導き出す ・まったくの新規プロジェクトの場合で運用実績がない場合はトラフィック量とパターンを見極めるのは困難なので、測定のためのプロトタイプを用意するか、本番移行前に、仮リリースしてトラフィックを測定する ・それも困難である場合は、前提条件を明確にしてトラフィック量・パターンを決めておき、それについて顧客と合意しておく
H72	組織	プロジェクト・マネージャは規模や性質に見合った経験を有するか、経験不足時に組織的支援策（プロジェクト責任者を含む）はあるか？	プロジェクト・マネージャのスキルと実際のプロジェクト規模・難しさをよく検討していること	・管理型 PMO ではなく、支援型 PMO になっているか？ ・火を噴く前に支援する文化があるか？ ・経営側の支援組織があるか？ ・最初は良いが、途中でおかしくなるケースもある	・プロジェクト計画書	・プロジェクト・マネージャに対して的確なプロジェクト・マネージャ教育を行う ・コーチを付ける ・プロジェクト計画書を有識者でレビューする ・PMO に協力を要請する
H73	組織	組織の責任・権限の明確度と組織機能（事務局など）は十分か？	工程別に、組織の責任・権限、会議体、目的を事前に定義しておくこと。 マネジメント・レベルの会議体も準備し、想定外の問題にも対処可能としておくこと	部門の責任範囲を明確にすることで、会議体への参加意識が高まり、施策のフォローが容易になる	・体制図 ・会議体	・体制図の見直しと責任範囲の明確な宣言をプロジェクト計画書などで文書化する

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H74	組織	協力会社が多階層になっており、責任範囲があいまいになっていないか？ 多階層構造に起因する問題（責任範囲、コミュニケーション）はないか？	役割分担、責任範囲を明確にできていること	多階層になっていると責任の所在がどこにあるのか分からなくなりやすい	・体制図 ・役割分担表 ・ヒアリング	・各社のキーパーソンを集めた会議体を設置する ・その中で各社の責任範囲の確認・見直しを実施し、定期的にコミュニケーションを行う
H75	基本動作	システム変更時にはバックアップを取るようにしているか？	例えば、以下のような観点で確認を行っていること ・設定ファイルを修正するときはバックアップを取っていること ・モジュールを新しいものに変更するときは元のモジュールのバックアップを取っていること ・システムのバックアップを取っていること ・データのバックアップを取っていること	モジュールの入れ替えや設定ファイルの入れ替えなど、すぐに環境を戻せるように工夫している必要がある	・リリース手順書など	ファイルの上書きをしない、データベースを変更する場合もＤＢバックアップを取るなどの手順を明確にする
H76	基本動作	プロジェクト内で使われる用語の意味について、用語集などの一覧表を作って管理し、顧客およびプロジェクト・メンバーに認識のずれがないようにしているか？	用語集を作成し、関係者間で共有できていること	使われている用語の意味を取り違えると設計ミス、作業ミスを起こすことがある	・用語集	・用語集一覧を作って顧客およびプロジェクト・メンバーとレビューを実施する
H77	基本動作	議事録やバグ票を正しい日本語で書けているか？	議事録やバグ票の内容を確認していること	正しい日本語で書けていることはプロジェクトの状況を把握する上での前提事項となる。 特に議事録やバグ票を読む相手のことを考えて、正しい日本語で記述できているかどうかは重要である	・議事録 ・バグ票	・議事録やバグ票の日本語に問題がある場合は、教育的指導を実施する ・また、悪い事例の修正例を公開・情報共有して、注意を促す
H78	モチベーション	社員の労務状況（労務時間、作業環境）は悪くないか？	労務時間などを基に判断するとともに、現場に残っているかどうかを目視で確認していること	問題がある場合、原因分析を行ない、適切な対応策（増員や人材配置の見直しなど）を実施する必要がある	・勤務管理表 ・職場見学	・労務状況を改善するために、増員、人材配置の見直し、作業環境の改善などを実施する

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H79	モチベーション	人員の離脱が多くないか？（心身における体調不良、一身上の都合など）	休みや離職の傾向を確認していること	離脱が多い場合はプロジェクトへの不信感・危機感による場合がある	・体制表	・離脱原因がプロジェクトへの不信感・危機感である場合は、プロジェクトの区切り（終わり）の時期について明言してあげることで改善する場合がある ・プロジェクト推進に関してプロジェクトの一体感や仲間意識を醸成することで、離脱を抑えることができる場合がある ・プロジェクト・マネージャを補佐するスタッフをつけて体制を強化する
H80	モチベーション	各メンバーは自分の範囲について責任を果たすという意識が希薄ではないか？ 自分が担当する範囲以外について、無関心な対応をしていないか？	自分の直接の部下だけではなく、まわり全体を見渡してメンバーのモチベーションを見ていること	プロジェクト・メンバーの発言内容や発言回数、成果物の質など、普段の様子から責任意識を持って作業をしているかどうか？	・進捗会議などの議事録 ・報告書 ・担当者へのヒアリング	・各メンバーに、自分自身の作業内容について自ら進捗報告をしてもらう ・報告を受けた課題に対して対策や具体的な指示を行う
H81	モチベーション	プロジェクト・メンバー間でプロジェクトに対する不信感を表している者はどの程度いるか？	プロジェクト・メンバーとコミュニケーションができていること	プロジェクトに対する不信感が大きい場合はプロジェクト全体の士気を下げることにつながる	・メール ・ヒアリング	・プロジェクト・マネージャを補佐するスタッフをつけて体制を強化する
H82	モチベーション	各プロジェクト・メンバーに、このプロジェクトでの個人成長目標を持たせているか？	各プロジェクト・メンバーと個人成長目標について話し合いができていること	各メンバーが成長するための目標を持てるかどうかは、モチベーションに大きく影響する	・要員育成目標	・プロジェクトとして各メンバーの育成目標を定めて、各メンバーとその内容について合意する
H83	課題管理	管理されている課題の対策予定日が過ぎたものに対して、是正を実施しているか？	課題管理表でウォッチできていること	対策予定日が大幅に過ぎているものに対しては何か大きな問題がある可能性がある。 担当者がアサインされていない、担当者の認識がない、課題が大きすぎて解決策が具体的なところまで落とし込めていないなど	・課題管理表	・是正処置を実施していない原因を突きとめる ・特に課題内容が大きすぎて、誰が着手すべきか判断できない場合はキーパーソンを集めて対応策を検討する必要がある ・大きな課題については、エスカレーションして対策を打つ

No.	知識エリア	チェック項目	評価基準	個別のヒアリング要領	エビデンス・確認方法	対策案
H84	課題管理	管理されている課題について、顧客との合意がとれているか？	顧客との課題レビューを実施し、議事録を作成し、その議事録について合意を得ていること	・課題管理では、顧客との合意が重要。例えば顧客側では課題と認識しているのに、開発側ではそう認識していない場合は後に大きな問題に発展することがある ・重要度や緊急度の認識のずれも問題になる	・課題管理表 ・レビュー記録 ・議事録	・課題管理表のチェックを行い、顧客とレビューする
H85	課題管理	複数の課題解決策に対し、明確な判断基準（メリット/デメリット）を基に優先順位を付けた上で、解決策を選択しているか？	例えば、以下のような観点で確認を行なっていること ・解決策を複数考え、論理的な比較を行って、より適切な解決策を選択するようにしていること ・解決策が単なる思い込みになっていないこと	・経験や勘に頼った判断は、大きな判断ミスをすることがある ・判断には根拠や基準が必要である ・判断の材料となる選択肢についても、十分に検討し、より良い判断であるかを常に意識していることが重要である	・課題管理表	・判断の根拠を明確にした上で判断する ・判断の根拠、優先順位を課題管理表に記録する

付録 3.プロジェクトにおける問題事象と対策事例

知識エリア

No 事象タイトル

事象

◆原因	◆対策	◆対症療法
プロジェクトがトラブルに陥った原因と経緯	問題解決のためにプロジェクトで実施した対策	そのとき実施した解決策
		◆再発防止策（教訓）
		上流工程で実施しておくべきだった対策

統合

1 中核SEの離脱

業務設計の中核となっていた協力会社のSEが、結合テストの直前に入院してプロジェクトを離脱。SEが担当していた部分の詳細設計書やソースコードがPCにもサーバーにもない

◆原因	◆対策	◆対症療法
業務設計チームのリーダーとして、チームの進捗報告を定期的に行っており、単体テスト工程までは特に進捗上の問題も無く報告されていたし、進捗状況に関する質問にも的確に答えていたので、この中核SEを信頼して成果物の確認をしていなかった。 また、発注元とはプロジェクト全体の契約形態（2次請けと3次請けの契約）が請負契約か作業委託契約かで揉めており、チームの報告を受けてその内容を確認しようとしたが、どちらかあいまいだった	・作業委託でも作業者の管理責任は受託側にあると認識。応援部隊を編成し、当該リーダーが実施する予定の作業を短期間で完了させた ・リーダーに社員を配置するとともにドキュメント・サーバーによる文書の一元管理体制へ移行した。この部分の結合テストは、後半に実施するよう再スケジュールした	・人員の投入（増員） ・構成管理（ドキュメント） ・再スケジュール
		◆再発防止策（教訓）
		・構成管理のルールを決めること ・作業分担を明確にすること

統合

2 仕様凍結の遅延

1カ月で仕様確定、1カ月で開発、1カ月でテストという3カ月工程で予定されていた案件が、仕様確定のための1カ月間、販売先と打合せができず、設計検討も含めて実質2カ月に工程が短縮された

◆原因	◆対策	◆対症療法
アプリケーション開発を受注し、開発終了が3カ月後で確定していた。アプリケーションを販売する会社へのヒアリングで仕様を確定するはずが、販売先からの受注遅延により仕様の打合せを始めたのが2カ月目に入ってからとなり、開発期間を1カ月短縮せざるを得なかった	・小規模なシステムだったため、運用イメージを基に1カ月でプロトタイプを一式用意した ・2カ月目には客先で、プロトタイプを基にパーツ単位で仕様確定に持ち込み、追加・変更・削除の依頼に対応しながら開発と仕様確定を同時進行させた	・再スケジュール（リリース時期） ・機能削減交渉
		◆再発防止策（教訓）
		・フィージビリティ・スタディ（実行可能性の検証） ・契約によって顧客を巻き込む（仕様確定するまで顧客と共同作業） ・分割契約にする ・委任契約にする

統合

3 特定機能の進捗遅れ、品質劣化

機能変更のプロジェクトにおいて、ある重要な機能の業務ルールがドキュメント化されておらず、そのソースコードを修正できない（品質が悪化する）。当時の担当SEは他部門へ異動していた

◆原因

ソースコードがスパゲティ状態（構造化されていない）で、保守できない。プログラムを作り直そうにも、業務ルールがドキュメント化されていない

◆対策

- 当時のSEを1カ月間引き戻し、新メンバーとペアでコーディング、テストさせる
- 業務ルールは都度ドキュメント化（ソースコードの作り直しは実施せず）
- 当該機能は2次リリースへ延期

◆対症療法

- 再スケジュール
- 分割リリース
- 業務を知っている人材を確保
- 共同作業によるスキル・トランスファ
- プログラムの影響範囲調査

◆再発防止策（教訓）

- ドキュメント化の徹底
- レビューの徹底
- コーディング規約の策定と順守

統合

4 検収時期に一括納品できず

契約書には仮納品の記載はないが、顧客から事前検証や研修のため、3カ月前に仮納品するように言われていた。だが、進捗が遅れてすべての業務を一括して納めることができなかった

◆原因

顧客との約束がプロジェクト全員に共有されていない上、プロジェクト・マネージャは業務単位で完成した順にシステムを提供することで検収や研修に対応できると勝手に考えていた

◆対策

- プロジェクトの進捗状況を顧客に説明し、顧客の検収、研修を開発状況に合わせて順次、実施してもらえるように依頼した
- 顧客の上長に対して文面でのお詫びを行うとともに、経験が豊富で実績あるプロジェクト・マネージャを投入し、顧客交渉に当たらせた
- 開発体制を強化するとともに品質管理チームを発足させ、遅れや品質の問題解決に当たらせた

◆対症療法

- 分割リリース
- プロジェクト・マネージャ交代
- 問題点の解決管理
- 品質測定

◆再発防止策（教訓）

- 顧客とのコミュニケーション
- 作業分担の明確化
- プロジェクト内部の情報共有

統合

5 パッケージとカスタマイズのソースコード範囲が不明確

納品時に顧客へソースコードを開示することになっていたが、その範囲が不明確になった

◆原因

パッケージ製品に対するカスタマイズとして開発を受注した。パッケージ製品のソースコードは公開不可で、カスタマイズ部分についてのみ顧客に開示する契約であったが、顧客要求に対応するために、カスタマイズがパッケージのコア機能にまで入ってしまい、顧客への開示範囲について不明確になってしまった

◆対策

手を入れた部分がカスタマイズであると定義して顧客と調整し、プログラム単位で顧客に開示。コアソースの流出を防いだ

◆対症療法

- 顧客調整（契約内容）

◆再発防止策（教訓）

- 事前の契約確認（コア部分への修正に対する開示）
- プログラムの構造と権利範囲の明確化（著作権とコードの所有権）

スコープ

6 テスト工程で設計仕様変更が多発

総合テスト工程に入り、画面設計、業務仕様、出力帳票などで設計変更する個所が多発し、納期遅れにつながった

◆原因

Webベースの新製品の開発で、新しいプロダクトの技術的な特徴、性能、制約条件などを正確・的確に把握するのが困難だった。業務仕様においては、ユーザー要件の検討項目を膨らませ過ぎて仕様凍結が遅れ、仕様変更がテスト工程まで発生した

◆対策

・プロトタイプの作成後にプロダクトの特性を見極め、計画値との差異を検証して対策を検討
・顧客要求事項に関して、コスト、納期、品質面を考慮したシステム化対象範囲を明確にし合意する
・仕様凍結時期を限定する

◆対症療法

・バグか仕様変更かの切り分け
・対応の優先順位付け
・仕様凍結時期の明確化

◆再発防止策（教訓）

・分割契約
・仕様凍結時期の明確化
・十分な事前調査（制約、機能など）

スコープ

7 開発側の運用イメージ不足と仕様説明不足

本番稼働後、Webメール・クライアントで文字化けが発生し、使いものにならないという指摘を受けた

◆原因

メール・ヘッダーやエンコード手法はメール・クライアントごとに異なる場合がある。作成する機能の前提条件で、ある種のヘッダーと文字コードに限定した仕様としていたが、顧客は多様な取引先からメールを受信するため、提示している仕様ではメール・クライアントとして使いものにならなかった。仕様書は承認されていたが、実際には顧客はその意味が理解できていなかった

◆対策

・顧客も仕様を承認している以上、仕様変更として対策を実施。対策期間は、旧システムとの並行運用時
・開発担当が入手可能なフリーのメール・ソフトの各バージョンで機種依存文字を含むメールを送受信し、顧客が使用する想定範囲で動作するよう、一通りの実装と動作確認を実施した

◆対症療法

・バグか仕様変更かの切り分け
・旧システムとの並行運用
・バリエーション・テストによる問題点の把握

◆再発防止策（教訓）

・現状システムの調査の徹底
・現状運用の調査の徹底
・十分な受け入れテストの実施

スコープ

8 顧客の機能要件説明能力不足と開発側の顧客ニーズ把握ミス

顧客による動作確認段階に入ってから、仕様に対する苦情が多発した

◆原因

実装した機能については仕様書に載せて説明していたが、実装していない機能については説明していなかった。顧客は旧システムの各種機能を踏まえた仕様になることを想定していたため、大幅な意識のずれが発生していた

◆対策

・顧客も仕様を承認している以上、仕様変更として対策を実施
・旧機能のうち、便利だと思っていた機能を利用者からヒアリングし、コストと開発期間で合意に達したものをすべて実装した

◆対症療法

・仕様変更交渉の実施
・追加実装機能の優先順位付け

◆再発防止策（教訓）

・役割分担の明確化（仕様確定者）

スコープ

9 業務設計の中心となる設計リーダーが倒れて復帰できない

業務を熟知し、プロジェクトの中心となっていた設計リーダーが過労で倒れた

◆原因

あらゆる問題が1人の設計リーダーに回されていた。その設計リーダーは、部下への仕事の振り分け方があまりうまくなかった

◆対策

・業務設計の中心となる設計リーダーがいなくなった場合は、そのリーダーが仕事を抱え込むタイプで状況がブラックボックスになっているか、あるいは仕事を共有するタイプで部下もある程度分かっているかによって対応が変わる
・前者であれば、設計リーダーがいなければ仕事を進められないため、根本的にスケジュールや体制を見直す
・後者であれば、サブリーダーをリーダーにしてチーム全体を上位スライドさせ、さらに人員強化で体制を組み直す

◆対症療法

・再スケジュール
・人員投入

◆再発防止策（教訓）

・工程ごとに成果物と課題管理
・業務勉強会の実施
・ユーザーとの打ち合わせに開発者を出席させる
・徹底した勤務管理
・作業分担の工夫

スコープ

10 大幅な工数増加と工程遅延

見積もりと比較して、実際には大幅に工数が増加し、工程遅延となった

◆原因

当初の見積もり（実績値とほぼ一致していた）では顧客の予算に合わず、顧客から圧縮を求められた。対象業務の知見に乏しく、受注しないと先がないとの危機感も加わり、顧客の「簡単にできる」という言葉に反論できず受け入れてしまった

◆対策

・根拠の乏しい顧客の工数圧縮要請に安易に迎合しない
・類似プロジェクトを調査し、知見を十分活用する

◆対症療法

・人員投入（工程回復を図る）
・顧客との価格調整

◆再発防止策（教訓）

・過去の類似プロジェクトを調査して見積もりの妥当性を評価

スコープ

11 大幅な工数増加による採算悪化

受注範囲のうち、初経験領域での大幅工数増によりプロジェクトの採算が大きく悪化した

◆原因

初経験領域の受注に際して、体制の構築に手間取り、具体的な作業内容の検討に十分な時間を割けなかった。その状態で見積もりを出し、請負契約をしたため、作業範囲が肥大化しても追加費用の交渉ができなかった

◆対策

・初経験領域での受注に関しては、受注範囲の明確化に十分な時間を割く
・仕様や作業範囲の不透明な場合、請負契約は極力避けて、委任契約といったリスクヘッジできる契約形態にする

◆対症療法

・再スケジュール
・顧客との価格再交渉
・人員投入

◆再発防止策（教訓）

・初経験に対するリスク管理を十分行う
・作業範囲が分かるまでは委任契約にする
・不透明要素が高い場合は請負契約はしない
・初経験の場合は委任契約にする
・システムの重要度を考えた契約にする
・顧客とのリスクの共有とそれに見合った契約にする

スコープ

12 大幅な工数増加と工程遅延

**顧客からの納期要求、機能の追加・変更要求に対して反論できず、
一方的に受け入れざるを得なかった結果、工数増加と工程遅延を招いた**

◆原因

仕様がはっきりしない状態で、過小見積もりしたため、プロジェクト・マネージャ、管理要員、チームリーダーの人数が足りなかった。そのため、十分な仕様変更管理ができなかった

◆対策

- サブのプロジェクト・マネージャを投入し、管理要員、チームリーダーを追加した
- 仕様の承認ルール、仕様追加・変更スキーム、費用精算ルールなど、顧客と必要なルールを取り決めた
- 障害管理、リリース管理、構成管理など、プロジェクト・チームが順守すべき規約・標準類を整備し、順守の徹底を図った

◆対症療法

- プロジェクト・マネージャ支援要員の投入
- 手順、ルールの確立

◆再発防止策（教訓）

- 手順、ルールを事前に定義しておく
- 顧客とのルール、手順の合意
- 仕様確定の顧客との合意

スコープ

13 納期に間に合わない

**初めての業務で、当初想定していたより業務が複雑だった。
そのうえ業務の有識者も少なく、約束した納期に間に合わないことが判明した**

◆原因

開発が進むにつれ、部門によるローカルなルールで業務を運営していることが明らかになり、当初の見積もりを大幅に超える開発規模になった。また、業務の有識者が少なく、業務を共通、統合、集約する提案を顧客にできず、顧客の要望も断れなかった

◆対策

業務が想定以上に複雑で多岐にわたるため、運用が回る最小の業務に絞り込んで開発し、その他の業務は順次優先順位を付けて開発することにした

◆対症療法

- 顧客に協力を依頼
- 有識者の確保
- 人員投入
- 分割リリース
- 再スケジュール

◆再発防止策（教訓）

- 初経験に対するリスク管理を十分行う
- 仕様打ち合わせ会議に、業務をよく知っている業務部門の人に出席してもらう
- 分割契約
- 顧客との協力体制の確立

スコープ

14 仕様があいまいなままシステムを提供

初めての業務だったので、顧客が承認した詳細設計に基づいて開発したが、サービス開始後に不具合が多発した

◆原因

部門で業務仕様に差があり、そのバリエーションは当初想定した以上だった。詳細設計書を承認した顧客も気づいていたが、途中で顧客側の担当者が変ってしまった。開発側も要員追加ができず、仕様を膨らまさないように、詳細設計通りで本当にいいのか、あえて確認しなかった

◆対策

・部門ごとに業務仕様が異なることを顧客にも認識してもらい、仕様をすべて調査するよりも、運用で不具合を見つけ出すことにした。不具合を発見したら仕様を確定し、システムに反映した
・不具合に対して、運用支援で対応するチームと仕様を確定させてシステムを修復するチームを作り、本運用で発生した不具合に応じて順次修正することにした

◆対症療法

・使ってみて不具合をあぶり出す
・ドキュメントを残す
・修正作業の体制整備

◆再発防止策（教訓）

・事前調査を十分に行って事前に仕様を明確にする

スコープ

15 プロジェクト・マネージャが多忙

プロジェクト・マネージャの健康維持が課題になった（倒れてしまうと代役がいない状態）

◆原因

プロジェクト・マネージャの能力、保有情報が突出し、過負荷となった

◆対策

プロジェクト・マネージャのまわりに人材を増やし、軽微な雑用から、使いはしり、そのほか代理可能な業務を積極的に引き受け、プロジェクト・マネージャの負荷軽減を図った

◆対症療法

・プロジェクト・マネージャ支援要員の追加

◆再発防止策（教訓）

・勉強会の実施
・作業分担の明確化
・役割分担の明確化

スコープ

16 共同受注案件での開発分担不明確による混乱

公募案件を共同提案で受注後、元請けとの分担が不明確となり混乱した

◆原因

受注優先で共同体制を組むが、受注後の役割分担、予算配分の詰めがなく、受注後に混乱した

◆対策

受注優先であっても、受注後の役割分担を文書で明確にし、合意しておく。予算配分、責任分担を明確にしておく

◆対症療法

・交渉（役割、費用）

◆再発防止策（教訓）

・役割分担の明確化
・予算配分と責任分担の明確化

スコープ

17 要件定義が不十分で、テスト工程になってリリース必須機能の漏れが発覚

**要件定義が不十分で、テスト工程になってからリリース必須機能がないことが発覚し、
当初予定していた日のリリースが難しい状況となった**

◆原因

過去に経験したプロジェクトと類似だと判断したが、実際はそうではなかった。現行機能について十分な理解をしないまま要件定義を実施したため、後工程で要求がたくさん上がり、工数が膨らんでしまった

◆対策

・開発体制を増強する。自社メンバーだけではなく、協力会社側にもメンバー増強を依頼
・スコープを限定して一部機能をリリース

◆対症療法

・体制強化と人員投入
・分割リリースの実施
・顧客との仕様の再確認

◆再発防止策（教訓）

・ドキュメント作成
・工程区切りの評価
・顧客によるレビュー
・事前調査を徹底する（過去の類似プロジェクトと本当に類似かどうか）

タイム

18 テストの進め方が分からず、テスト計画もできない

**総合テストに入ったが、いままで小規模の開発しかやっていないメンバーばかりで
大規模開発での総合テストのやり方を全く理解していなかった**

◆原因

大規模開発の経験不足、知識不足

◆対策

・テストの進め方が分からない場合は、社内の類似の別プロジェクトのものを流用する
・大規模な総合テストを実施したことのある経験者を連れてきて計画を立てる

◆対症療法

・有識者の確保
・ルール、手順の入手（経験者による判断が必要）

◆再発防止策（教訓）

・組織プロセスの定義（プロジェクト・マネージャの任命方法、要員、体制の確立など）

タイム

19 稼働1カ月半前でも結合・総合テスト計画が不十分

稼働1カ月半前になって、事業責任者（プロジェクト・マネージャの上長）から、「プロジェクトが危なそう」というアラームが出た。単体テスト中なのに、結合テスト、総合テストの計画が立っていなかった

◆原因

元請け会社のプロジェクト・マネジメント不足、要員不足。開発委託先（3社）の責任範囲は、社内で行う単体テストまでであり、結合テスト（社間テスト）以降は、元請け会社が取りまとめる責任を負っていた

◆対策

- 委託先を含めて、メンバー全員を1カ所に集結
- 元請け会社の体制強化。委託先のキーパーソンを元請けの一員としてテスト計画を立案
- 毎朝、プロジェクト・マネージャと実務責任者のミーティングを実施
- 1次リリースと2次リリースに分割（一部機能はリリースを延期）

◆対症療法

- メンバーの集結と一体感の醸成
- PMOの投入
- テスト計画とテスト環境整備
- 再スケジュール
- プロジェクト体制の強化

◆再発防止策（教訓）

- 計画重視型のプロジェクト推進
- プロジェクト管理体制の整備
- 工程区切りでの評価

タイム

20 大幅な工数増加による採算悪化

移行作業に関して、大幅な工数増によりプロジェクトの採算が大きく悪化した

◆原因

移行作業に関して作業量を絞りきれないまま、顧客の要請に応じて請負契約で受注。実際には工数が予定より大幅に増加した。受注に際して前提条件を付けたものの、明確ではなかったため、追加費用の請求に応じてもらえなかった

◆対策

仕様や作業範囲が不透明な場合、請負契約は極力避けて、委任契約といったリスクヘッジできる契約形態にする

◆対症療法

- 顧客との価格再交渉
- トップ会談での価格調整

◆再発防止策（教訓）

- リスクに応じた契約形態
- リスクの高いことを相手に伝えて、理解してもらえない場合は受けない

タイム

21 共通ライブラリのスケジュール遅れ、品質不良

アプリケーションの開発と並行して、共通ライブラリを開発していたが、共通ライブラリの開発が遅れ、アプリケーションの開発スケジュールにも大きな影響が出た

◆原因

クリティカルパスである共通ライブラリの開発スケジュールの管理不良と、機能要件の定義不足（アプリケーションと共通ライブラリの機能分担の責任があいまい）

◆対策

- 共通ライブラリのマニュアル整備（機能要件の定義を追記）と開発体制の強化
- アプリケーションの単体テストはスタブで実施

◆対症療法

- 人員投入
- 設計の見直し
- テスト実施方法の工夫

◆再発防止策（教訓）

- ハイスキル要員の確保
- クリティカルパスを意識した進捗管理
- 工程管理基準の定義

コスト

22 見積もり時の前提想定不足

顧客先での立会いや運用サポートのSE作業が大量に発生し、想定以上のコストがかかった

◆原因

契約時に作業場所や作業内容が明確化されていなかった。そのため、交通費は見積もりに入っていないが見積もり範囲外と宣言できなかった。SE作業で依頼される範囲が不明確なため、依頼された作業量が予定コストを超過しても、見積もり範囲外と宣言できなかった

◆対策

顧客調整により本契約範囲内での作業範囲を明確化し、依頼を受け続ける体制から作業終了目標をはっきりとさせるようにした

◆対症療法

・顧客調整
・作業範囲の明確化

◆再発防止策（教訓）

・契約作業範囲の明確化
・他プロジェクトの見積もり例の参照

品質

23 メーカー提供プロダクトの不具合でシステムが動かない

メーカー提供のプロダクトを大規模に利用したが、メモリー・リークが多発して実用に耐えなかった

◆原因

プロトタイプを作成して、技術的な見極めを完了したと思っていたが、開発で本格的に利用してみるとメモリー使用量が徐々に上がっていき、最後はオーバーフローしてしまった。大量データ登録での確認不足が原因だった

◆対策

この状況における対策は、何としても解決するか、早くあきらめて代替機能を探すことに重点を置くかのどちらか。見込みがない、あるいは解決できる自信がない場合は早くあきらめることが肝心。パッケージの調査をして、対応方針を検討し、顧客と調整する

◆対症療法

・ベンダーを巻き込む
・パッケージの選び直し
・スクラッチで作り直す
・スケジュール再調整
・運用回避検討（日次のシステム再起動など）

◆再発防止策（教訓）

・保守契約を結んでおく（保証の範囲を明確にする）
・パッケージの調査を事前にしっかりやっておく（実績を調査する、機能の網羅性、事前のテスト）
・パッケージが顧客からの指定の場合は、責任範囲を明確にする
・事前に実績のあるパッケージを調査・準備しておく
・パッケージの使用経験者を確保する
・パッケージ提供会社の調査（安定性）

品質

24 システムの品質が悪い

**スケジュールに追われ、低品質のものをリリースし、
障害対応でドタバタする、という負のループから抜け出せない**

◆原因

品質の作り込み（ドキュメント作成、レビュー）や十分なテストよりも、スケジュール/コストを優先していた。開発担当者が、当該システムで求められる品質に対する認識の甘さがあった

◆対策

・品質管理を中心に管理要員を追加投入
・リリース・スケジュールに関するルール、障害対応ルールなどを整備し、その順守を周知徹底した

◆対症療法

・品質分析の実施（弱点部位の特定）
・人員投入

◆再発防止策（教訓）

・品質要求の明確化
・テストの十分な実施
・リリース品質基準の明確化
・品質に関して顧客と共通認識を持つ

品質

25 システムの品質が悪い

結合テスト時、単体テストレベルのバグが多数発生。結合テストを中断・延期し、工程遅延を招いた

◆原因

追加機能を開発していた際、メンテナンス性の向上を図るため、既存プログラムの修正も実施。その修正部分のテストを十分に行わないまま結合テストを始めてしまった

◆対策

・プログラムの単体テストを再実施
・第三者によるソースコード・レビューを実施

◆対症療法

・テストの再実施
・ソースコード・レビューの実施

◆再発防止策（教訓）

・機能追加ルールの明確化
・機能追加のリスクを顧客と合意

品質

26 大量印刷時間が長い

**クライアント/サーバー・システムで開発した業務パッケージをWeb化した。プリント・サーバーの
印刷処理方式を自社で構築したが、総合テストの大量印刷処理で想定を超える処理時間がかかった**

◆原因

Webアプリケーション・サーバーとプリント・サーバー間のデータ受け渡し処理で、ミドルウェアの共通資源が不足し、処理待ちが発生していた

◆対策

ミドルウェアの共通資源の設定を見直すとともに、新たに大量印刷の処理方式を構築し、印刷量の多い業務は新しい方式に変更した

◆対症療法

・類似個所の見直し
・処理方式の見直し

◆再発防止策（教訓）

・事前調査（技術調査）
・事前の負荷テストの実施
・性能要求の検討と明確化

品質

27 ユーザーの研修で急に処理が遅くなる

ユーザーの業務運用研修で、同じ業務処理を同時に行うと処理時間が大幅に遅くなる

◆原因

認証サーバー、DBサーバーへの処理が集中したうえ、共通資源の排他制御の不具合、SQL文の使い方の誤りによる無駄なDBアクセスが多かった

◆対策

- 共通資源の排他制御方式を見直し、トランザクションが集中する業務プログラムを優先して修正
- DBアクセス・ログを解析し、特に無駄な処理を行っている業務プログラムのSQL文を修正

◆対症療法

- 専門家を連れてくる（方式の見直し）

◆再発防止策（教訓）

- 事前調査（技術調査）
- 有識者による設計レビュー
- べからず集の作成

品質

28 発注時の丸投げによって失敗

業務知識・構築実績があると思われた協力会社に開発を委託したが、進捗・品質に問題が発生した

◆原因

協力会社のマネジメント力の不足。また、協力会社を信頼して全面委託する場合、協力会社からの見積もりの検証や品質チェックが不足していた

◆対策

- 協力会社の開発体制を強化する
- 顧客からの要件がまだ増える場合には、顧客との調整でスコープを整理する
- 協力会社に対するプロジェクト・マネージャ支援、作業支援を通して、プロジェクトの一体感を醸成する

◆対症療法

- 協力会社の体制強化
- プロジェクト・マネージャ支援要員の追加
- 作業分担の見直し

◆再発防止策（教訓）

- 協力会社の事前の調査・評価
- 協力会社のプロジェクト管理の徹底
- 協力会社とのコミュニケーション計画の合意（進捗報告の義務など）

品質

29 パッケージのバグフィックスに時間がかかる

パッケージの機能のうち、初めて利用した機能の品質が悪く、そのバグ修正に時間を要した

◆原因

パッケージの機能のうち、初めて利用した機能の品質が悪かった。海外のパッケージ・ベンダーだったこともあり、バグ対応に時間がかかった。また、バグ修正により機能がデグレードしたこともあり、受け入れテストを十分に実施する必要があった

◆対策

- パッケージの中で初めて利用する機能は、その品質を十分確認しておく
- バグがあることを前提としたテスト・スケジュールを立てる

◆対症療法

- 迂回策の検討
- 代替製品の調査
- アドオンによるカスタマイズ対応

◆再発防止策（教訓）

- テストスケジュールの作成
- パッケージのサポート体制の確認
- サポートに対する覚え書きをとる
- 事前調査（初利用パッケージの機能・性能）
- リスクの顧客との共有

品質

30 大量データ処理方式の設計ミス

**総合テストに向け、テスト環境とテスト計画を策定したが、
アプリケーションの開発リーダーから性能問題に関する懸念があがった**

◆原因

現行システムの業務集中と処理能力の改善報告を受け、アプリケーションの開発リーダーから新システムの処理方式に問題提起があった。方式開発リーダーは業務量に応じた性能設計を行い、事前にテスト環境で確認していたが、業務運用に伴うトランザクションの傾向には疎く、想定外であった

◆対策

・処理方式を見直す
・業務処理の集中を避けるための運用制限をかけるべく、本番環境を用いて性能テストを実施し、顧客の承認を得た

◆対症療法

・類似個所の見直し
・運用制限
・処理方式の見直し

◆再発防止策（教訓）

・事前調査（運用特性）

品質

31 あいまいな仕様のまま実装し、テスト工程で要件不一致が多発

開発委託先からの納品を受けて結合テストを開始したが、要求機能を満たしていなかった

◆原因

要件（業務ルール）のドキュメント化が不十分で、プロジェクト・マネージャ(元請け会社)の頭にしかなかった。開発委託先の成果物に対するレビュー不足

◆対策

・テスト手順書をディジョン・テーブル形式で記載
・そのテスト手順書に基づいて、開発委託先に再度プログラミングと単体テストを依頼
・1次リリースと2次リリースに分割

◆対症療法

・業務ルールのドキュメント化
・テスト手順のドキュメント化
・再テストの実施
・コードレビュー
・再スケジュール

◆再発防止策（教訓）

・仕様のドキュメント化

品質

32 外字印字の不良

文書を印字するシステムで、本番稼働後に、文字が抜けて空欄のある帳票が印刷されていた

◆原因

学術系の文字が外字として登録されていたが、開発したシステムからこれらの外字を印刷するためには、運用環境でも印刷用に外字の設定が必要だった。しかし、外字登録の範囲や必要性を開発側が把握しておらず、外字のコードをそのままプリンタに送ったため、問題が発生した

◆対策

外字設定を実施後、空欄が発生していないか印刷結果に対して全件目視確認を実施した

◆対症療法

・全文字確認テストの実施
・顧客の要求仕様の明確化

◆再発防止策（教訓）

・事前調査（現行調査）

品質

33 運用中の業務停止

テスト環境では問題なかったが、運用開始後にシステムが異常終了してしまった

◆原因

テスト環境と本番環境が異なるため、本番環境への移行時にパラメータを修正する必要があった。本番環境での動作確認はできないため、パラメータ・ファイルの差分を抽出して確認した。ただし、「正しい差分が出ていること」はチェックしたが、「差分が出ないということは反映もれ」というチェックを怠ったため、テスト環境のパラメータが残ってしまった

◆対策

全ソースコードのパラメータ・リストを出力し、本番環境とテスト環境の差異を把握している有識者2人で目視確認した

◆対症療法

・設定状況の見直し
・有識者の参画
・構成管理者を立てる

◆再発防止策（教訓）

・有識者によるレビュー
・テスト計画のレビュー
・リリース手順の見直し

人的資源

34 指揮・命令系統が不明確

プロジェクト内の指揮・命令系統にあいまいな部分があり、プロジェクト・マネージャによるリソース調整がうまくできなかった

◆原因

このプロジェクトは、専任のAチームと他のプロジェクトを掛け持つBチームで構成していた。Bチームに対するプロジェクト・マネージャの権限があいまいであったため、プロジェクト・マネージャはBチームに対するリソース調整ができなかった

◆対策

プロジェクト推進における責任分担、指揮・命令系統の明確化

◆対症療法

・体制俯瞰図の作成と見直し
・作業分担の明確化
・責任の明確化
・チーム間の作業の調整（優先順位の判断、プロジェクト・マネージャやその上司との意識合わせ）

◆再発防止策（教訓）

・プロジェクト計画時から責任分担と作業分担を明確にする
・作業負荷の明確化
・プロジェクトのスコープを明確化

人的資源

35 プロジェクト編成の中でプロジェクト・マネージャ的な人材が2人存在する形になり混乱

営業活動から参画していた全体的視野を持つプロジェクト・マネージャのもとに、開発部隊を率いた実力派の開発リーダーが参加。その実力ゆえにプロジェクト・マネージャが二重に存在するような状態となった

◆原因

社内で、開発の統制に関する仕切りが不十分だった

◆対策

開発チーム立ち上げ時の社内統制に細心の注意を払う。上級管理職、幹部の監視と振る舞いが重要

◆対症療法

・責任分担を明確にする

◆再発防止策（教訓）

・プロジェクト計画時から責任分担と作業分担を明確にする

人的資源

36 他業種から転進した元請け会社が未熟で混乱

他業種から転進した元請け会社のもとで2次請けするが、元請けの作業標準が確立しておらず、元請けを教育しながらの開発となった

◆原因

他業種からの転進してきたため、元請け会社としての技量を容易に獲得できず、未熟なスキルのまま元請け業務を推進していた

◆対策

・元請け会社、2次請け会社の機能分担を明確にし、長期的な企業連携による技量向上を図る
・2次請けは元請けにソフト作業標準を教え、基本的な手順について合意を形成

◆対症療法

・作業分担の明確化
・開発標準の共有化

◆再発防止策（教訓）

・開発標準に関する合意

人的資源

37 DBMSの技術的な問題で対応できない

DBMSに不慣れな開発要員だけでアプリケーションを開発。DB操作のレスポンスが遅すぎて、システムとして使い物にならなかった

◆原因

ITアーキテクト要員の不足。本来なら利用技術が確定したところで必要な技術者をアサインし、技術グループを構成するところだが、それをしないで進めてしまった

◆対策

・時期的に遅くなっても専門の技術者を連れてこないと解決しない
・技術者にまかせきりにしないで、アプリケーション開発要員の数人を同じチームにして技術グループを構成する

◆対症療法

・技術者の確保
・開発要員への教育
・再設計
・再構築

◆再発防止策（教訓）

・有識者の参画
・要員のスキルレビュー
・要員教育

人的資源

38 結合テストで品質不良が発生

Webアプリケーション・サーバー上の結合テストに着手したが、プログラム設計の誤りによるバグが多発した

◆原因

詳細設計工程で業務仕様が増加したため、プログラム設計工程から多数の要員を追加した。これらの要員は、対象業務も、そのプロジェクトの開発環境も初めての経験だったうえ、進捗の遅れも重なり、オブジェクト指向によるプログラム設計や新処理方式に関する事前研修が不十分だった。研修不足を補うために有識者のレビューを実施することもなく、途中参加の要員にプログラム設計を任せていた

◆対策

・結合テストを中止し、対象業務の経験者を投入
・レビューを通じて発見した誤りを全員に周知して、各自がプログラム設計を見直し、全プログラムを改修した

◆対症療法

・再スケジュール
・経験者の投入
・ソースコード・レビュー
・追加要員への教育

◆再発防止策（教訓）

・研修の実施
・コーディング規約の策定

人的資源

39 受注後の要件詳細化、基本設計体制に不備

メンバーの経験が設計後の開発ばかりだったため、要件定義や設計能力に欠けていた。設計工程から遅延し始めた

◆原因

設計後の開発ばかり受注していたため、メンバーに要件定義や設計の能力が育たなかった。設計待ちの姿勢となった

◆対策

短期的には、必要なスキルを備えた要員を探し、投入する

◆対症療法

・経験者の投入

◆再発防止策（教訓）

・設計者の育成
・上流設計技術者の採用

人的資源

40 大幅な工数増加と工程遅延

人的リソースを十分に確保できていない状態だったが、要員不足・工期不足にも「なんとかなる」といった甘い判断でプロジェクトを進めていた

◆原因

プロジェクトに必要なスキルを持った人的リソースが、他のプロジェクトに取られてしまった。その補充要員はスキル不足で、従事可能期間もかみ合わず、結果的に数少ないキーパーソンに負荷が集中。機能横断的なコーディネート、設計不備などのチェック、要件拡大に対する適切な対応がタイムリにできなかった

◆対策

・サブプロジェクト・マネージャの投入
・管理要員の投入
・開発要員の投入
・残タスク（機能開発、障害対応、仕様変更対応など）の洗い出しと、そのタスクの実施スケジュールの作成、担当者のアサイン、顧客との調整を実施
・日次での進捗管理

◆対症療法

・組織マネージャへのエスカレーション
・人員投入
・残タスクの洗い出し
・再スケジュール
・日次での進捗管理
・顧客調整（プロジェクト間の全体最適）

◆再発防止策（教訓）

・リソースの十分な確保
・リスク管理
・プロジェクト・マネジメントの徹底

人的資源

41 業務担当リーダーの離脱

業務には詳しいがマネジメントでは不安があった業務担当のリーダーが、総合テストの直前にプロジェクトを離脱

◆原因

総合テストを控え、業務に一番詳しい人を業務担当リーダーとしてマネジメント・チームに加えるよう指示があった。業務の有識者はいたが、協力会社の人でもあり、マネジメントには不安があった。プロジェクト・マネージャは別の人をアサインするよう提案したが、方針ということで断られた

◆対策

・マネジメントに長けた経験者をチームに投入
・業務の有識者がいなくなったので、残ったメンバーとの面談を通じ、業務仕様を任せられる人を選定
・顧客に協力を依頼し、業務仕様のレビューを行いながら人材育成する

◆対症療法

・経験者の投入
・業務有識者の投入
・業務担当リーダーの育成
・顧客への協力依頼

◆再発防止策（教訓）

・マネジメント能力のあるプロジェクト・マネージャの任命

人的資源

42 協力会社のリーダー、中核要員がくるくる変わる

プロジェクトの立ち上げ時に協力会社の要員が交代する

◆原因

好況時、人材が出払っているにもかかわらず、協力会社が営業的に受注を優先し、場つなぎの人材を当てて、急場をしのいでいた。本格的な体制が組めるようになったのは、プロジェクトがかなり進んだ段階だった

◆対策

・人材派遣型契約の抑制
・ドキュメント主義型の開発スタイルをとる

◆対症療法

・本格体制の構築
・再スケジュール

◆再発防止策（教訓）

・仕事に応じた体制の構築

人的資源

43 協力会社のスキルが低すぎてまったく役に立たない

協力会社を使って設計をさせたが、まともに設計ができない。内容のレベルが低い

◆原因

協力会社の要員のスキル不足、経験不足

◆対策

プロジェクトの状況によるが、本来は協力会社を替える必要がある。ただし、ポテンシャルがあれば、教育することで良い協力会社に変わることができる。協力会社に限らず、スキルを持っている要員を集められればよいが、実際はスキルのない要員も多いので、結局は教育をした方が得策なこともある

◆対症療法

・協力会社要員への教育実施
・設計レビューの実施
・業務有識者の投入

◆再発防止策（教訓）

・協力会社の能力の調査
・協力会社へのガイドラインの作成

コミュニケーション

44 プロジェクト・マネージャ、リーダーと実務担当者の関係が悪化してコミュニケーションも作業も進まない

次期システムの構築に当たり、実務担当者は旧システムの保守から携わっているため業務知識が豊富である。一方、プロジェクト・マネージャとリーダーは別のプロジェクトから引き抜かれた要員である。やり方の違いなどで意見が合わず、コミュニケーションも悪化している

◆原因

やり方、進め方、コミュニケーションのとり方のミスマッチがあった。問題の発見が遅れた

◆対策

- この問題はプロジェクト責任者でないと対応できない
- プロジェクト・マネージャか担当者のどちらに問題があるのかを明確にして、問題のある方を是正する

◆対症療法

- 関係者からの状況ヒアリング（個別面談）
- 議論の場を作る（合宿の実施など）
- 実務担当者からの説明会実施
- お互いの仕事の状況を情報共有
- 課題の情報共有

◆再発防止策（教訓）

- 作業方針を示す
- 会議体の設定
- キックオフの実施（プロジェクト・マネージャが変わったらキックオフをし直す）
- プロジェクト・マネージャのメンバリングとチームビルディングの違いを認識する

コミュニケーション

45 役割分担が不明確

アプリケーション開発担当チームと運用担当チームのコミュニケーションが不足して、手戻りが発生。作業負荷が大幅に増えて、メンバーが疲弊した

◆原因

アプリケーション開発担当チームと運用担当チームが連携してプロジェクトを進めるのは初めてで、双方の役割分担が不明確だった。両者間の情報伝達も悪く、特にアプリケーション開発担当チームから運用担当チームへの情報伝達の遅れや度重なる仕様変更により、運用設計の作業負荷が増大した

◆対策

- それぞれの分担と責任範囲、連携方法を明確にする
- 各チームの定例ミーティングに双方のリーダー、キーパーソンが出席して情報交換する

◆対症療法

- メンバーを集めてミーティングをする
- チームビルディング
- ルール作り

◆再発防止策（教訓）

- コミュニケーション・ルールの策定
- 作業計画の情報共有

コミュニケーション

46 複雑な開発組織内で、口頭による依頼、周知が多用され混乱

開発組織が複雑で、作業標準が統一されておらず、口頭による連絡が多用されて大混乱となった

◆原因

複雑な組織構成の中でコミュニケーションを文書化しておらず、コミュニケーションの管理が不在となった

◆対策

・必ず文書でコミュニケーション（依頼、回答、周知など）をとる
・この文書（1件1ページ）に通番を振り、通番管理をする
・これら文書の管理者を置く。関係者の進捗管理会議で、この文書管理を実施する

◆対症療法

・問題点一覧の作成（情報集約）
・コミュニケーション・ルールの策定
・メーリングリストの作成
・伝達事項の文書化

◆再発防止策（教訓）

・コミュニケーション・ルールの策定
・問題点管理ルールの確立

コミュニケーション

47 事務方との連携が悪く、開発グループが客先で孤立

2次請けしたプロジェクトで、元請け会社内に詰めている開発グループの労働環境が悪化し、疲弊した

◆原因

2次請け会社の事務方が開発現場の状況を把握できず、労務的に放置状態となっていた

◆対策

・事務方を開発現場に連れてゆき、状況を把握させ、会社として対応可能な施策をすべて打たせた
・ホテルの手配、タクシー券の配布、補給食の手配
・事務的サポート要員の手配、通信環境の改善、予算措置など
・随時、現場の状況を把握する体制を確立した

◆対症療法

・作業環境の改善
・PMOの派遣
・事務支援要員の投入

◆再発防止策（教訓）

・事前の作業環境計画

コミュニケーション

48 総合テストが品質不良で進まない

結合テストまでは先行していたチームにおいて、総合テストで品質不良が判明した

◆原因

結合テストまでの進捗が一番早かったチームから総合テストに入ることになった。そのチームの業務担当リーダーは詳細設計以降の仕様変更・追加を把握しており、テスト範囲を限定しながら総合テストを進め、並行して仕様変更・追加することを考えていた。しかし、その考えが顧客に伝わっておらず、総合テストは顧客の都合に合わせてスケジュールされていた。仕様変更・追加への未対応部分がすべて品質不良となった

◆対策

・総合テストの開始を遅らせ、仕様変更・追加の開発を先に実施した
・業務担当リーダーは総合テストの実施に当たり別の作業場所に移ったので、新たに業務知識のあるサブリーダーを選び、仕様を整理させた
・サブリーダーは定期的にマネージャとコミュニケーションをとり、業務仕様の確認作業を進めた

◆対症療法

・再スケジュール（テスト開始時期）
・業務担当リーダーの育成

◆再発防止策（教訓）

・顧客との進捗状況の共有
・テスト計画の顧客との合意

コミュニケーション

49 仕様書と実装の差異に顧客がクレーム

基本設計書に基づいて作成した画面を顧客に確認してもらったところ、画面内の微妙な差異について指摘を受け、ドット単位で修正することになった

◆原因

画面仕様書はプロトタイプを基に作成したが、顧客はそれが完成イメージであると認識していた。実装後にある程度の差異が発生することを事前に説明していなかったため、顧客との調整に失敗した

◆対策

仕様書に完全一致するよう、全画面を再修正した

◆対症療法

・要求に合わせて修正を実施

◆再発防止策（教訓）

・厳密性を要求される画面・帳票イメージの顧客確認

リスク

50 顧客側のプロジェクト・マネージャが交代してしまった

設計が完了し、これから開発に入るという段階で、顧客側のプロジェクト・マネージャが突然会社を辞めてしまい、プロジェクトの運営に支障を来たした

◆原因

事情はどうあれ、顧客側のプロジェクト・マネージャが交代することはあり得ることである。そのときの対策を速やかに打たなかったため、プロジェクトの運営に支障を来たした

◆対策

・プロジェクト・マネージャが社内でどの程度状況や進捗を報告しているかを確認する
・新しいプロジェクト・マネージャを配属してもらうよう依頼する
・新しいプロジェクト・マネージャが配属になったら、すぐに集中ミーティングを実施して、プロジェクトの概況、進捗状況、予算と実績、体制、リスクなどを共有する
・新しいプロジェクト・マネージャは最初に情報を提供してくれる者を信用する傾向があるので、情報を提供する側は早さと正確さを心がける

◆対症療法

・新しいプロジェクト・マネージャの指名の依頼
・プロジェクト・マネジメントの方針を新プロジェクト・マネージャに伝える
・キックオフの実施
・議論の場を作る（合宿の実施など）
・実務担当者からの説明会実施
・お互いの仕事の状況を情報共有
・課題の情報共有

◆再発防止策（教訓）

・リスク管理
（情報の文書化、顧客側のサブリーダーの確保）
・契約書にプロジェクト・マネージャの交代に関する制約事項を記載する
・キーパーソン、プロジェクト・マネージャの交代時には事前に連絡をもらうようにしておく

リスク

51 協力会社が設計を完了した時点で次工程の開発を断ってきた

**基本設計を委託していた協力会社が設計完了間近になって、
「基本設計までは請け負うが、次の工程からは撤退したい」と通告してきた。意思はかなり固い**

◆原因

協力会社は、プロジェクト・マネジメントへの不安、設計内容を実現することへの不安、次工程で要員増加できない不安などを抱えていた

◆対策

・まずは引き止め交渉をする。それが無理となったら、次の協力会社への引き継ぎを確約させる
・その上で、コストも含めた引き継ぎ策を協力会社と決める（通常、引き継ぎにかかるコストは支払わない約束を取り付ける）

◆対症療法

・引き留め工作
・引き継ぎ作業の確約と契約

◆再発防止策（教訓）

・マネジメントの強化
・協力会社との課題共有
・協力会社の能力に応じた範囲での発注

リスク

52 業務有識者不在での総合テスト実施

**本番環境で総合テストを実施する前に担当者が交代した。急いで別の担当者をアサインしたが、
顧客との調整やテスト・シナリオの作成に想定以上のコストを費やすこととなった**

◆原因

総合テスト開始前に業務有識者が離職し、新しい担当者が業務知識や顧客対応の面で先任者のレベルに追いついていなかった

◆対策

プロジェクト・マネージャと関連業務を担当している有識者を招集して臨時に対応した。1人で実施する予定だった作業に人数をかけたので、コストは人数分増加したが、顧客に対する影響はほぼ解消できた

◆対症療法

・次のキーパーソンを育てる
・再スケジュール
・ドキュメントがあるかのチェック
・有識者を連れてくる

◆再発防止策（教訓）

・キーパーソンを複数人置く
・ドキュメント化する（仕様書だけでなく、判断基準、業務ノウハウなどを明記する）
・受託時によく検討する（業務に詳しい人が複数人いること）

調達

53 契約前に作業完了

短期の小規模システム開発において、契約未締結のまま開発を開始し、工期完了まで実施してしまった

◆原因

開発計画と見積もりに対し、顧客から承認の連絡を得て開発作業を開始した。月内に内示をもらったが、正式発注は工程期間内に実施されなかった

◆対策

顧客側担当者の上長を含めて交渉の機会を設けてもらい、正式発注手続きを早急に実施しなければ、システムを納品できない旨を伝えた。翌月、正式発注された

◆対症療法

・正式発注手続きを強く求める

◆再発防止策（教訓）

・発注をもらってから開発に着手

調達

54 ベンダー提供の業務パッケージの品質不良による運用の中止

パッケージ・ソフトをカスタマイズしてシステムを稼働させたが、品質と性能の問題が生じ、稼働後すぐに運用を中止することになった

◆原因

パッケージの仕様として提供されるものと、カスタマイズして開発するものとの判別を、パッケージ・ベンダー任せにしていた。これが原因でカスタマイズの開発期間を十分に取れないまま、納期を迎えてしまった。また、プロジェクト・マネージャが顧客やパッケージ・ベンダーとの仕様調整に悩んで離脱。交代したプロジェクト・マネージャは兼務だったため、プロジェクトの状況や問題を十分に把握できないまま、パッケージを導入して稼働させてしまった

◆対策

・パッケージ・ベンダーや業務有識者による状況判断の結果、品質・性能を改善するために稼働を半年遅らせることにした
・パッケージ・ベンダーとの開発分担を見直し、開発体制を立て直した

◆対症療法

・現状調査の実施
・再スケジュール
・顧客交渉
・役割分担と開発体制の見直し

◆再発防止策（教訓）

・事前調査
・負荷テスト
・パッケージ・ベンダーとの責任分担の明確化
・第三者による進捗状況管理

調達

55 営業時に営業SEが決めたブラウザが実用に耐えなかった

受注活動の中で営業SEがブラウザの製品を決めて提案し、そのままシステム構成に組み込まれたが、テスト段階で実用に耐えないことが判明した

◆原因

営業SEは、製品の実用性を確認する立場になく、確認する組織が社内に存在しなかった。システム開発部門は、受注後、与えられた条件でシステム構築するだけだった

◆対策

・受注後、開発部門でシステム・アーキテクチャを再検討し、実現性の面から再評価、再設計する
・営業活動から引き継いだ条件を、無条件に引き継がない
・そのためのコストを見積もりに盛り込む

◆対症療法

・利用可能な代替製品の調達

◆再発防止策（教訓）

・事前調査

調達

56 営業時に営業SEが決めたRDB製品の最新版を自社ミドルウェア製品がサポートしていなかった

受注活動の中で、営業SEがRDB製品の最新版の利用を決めて提案し、受注後、そのままシステム設計が進んだ。しかし、これと連動する自社ミドルウェア製品がRDBの最新版をサポートしておらず、その対応に時間と費用がかかることが判明した

◆原因

営業SEは自社ミドルウェア製品の開発計画を正確に把握していなかった。RDBの最新版のサポートに要する費用、期間に関する正確な認識を持っていなかった

◆対策

・営業SEが自社製品の開発に関する正確な情報を保持できるような社内体制を確立する
・営業活動のシステム提案を社内で評価できる体制、そのために必要なコストをプールする体制を作る

◆対症療法

・調達品に関する顧客との調整

◆再発防止策（教訓）

・事前調査（パッケージ、社内製品の整合性など）
・営業社員への自社パッケージの教育
・提案内容のシステム担当者によるレビュー

調達

57 パッケージ・ソフトの利用によりブラックボックス化が問題に

**過去に実績のあるパッケージ・ソフトを導入したが、その品質が著しく悪かった。
パッケージ提供会社は「問題ない」と回答してくるが、実際の状況とかけ離れた内容だった**

◆原因

パッケージ・ソフトは内部に踏み込んだテストで品質を確認できないため、パッケージ提供会社からの回答をうのみにしていた

◆対策

パッケージ提供会社に品質管理体制やサポート体制の強化を依頼する。テスト支援などを依頼することもある

◆対症療法

・パッケージ・ベンダーを巻き込む
・アドオンによる回避
・代替製品の調達

◆再発防止策（教訓）

・パッケージ会社との保証契約

基本動作（拡張）

58 結合テストで重複障害やデグレートが多発

**結合テストに入り、バグを発見して修正しても、再度同じバグが発生したり、
インタフェース・ミスが多発したりする**

◆原因

バグ修正を複数のプログラムに対して実施した場合に、修正履歴、プログラムの世代管理が厳密に行われていなかった。未修正のプログラムを使ったり、担当者が勝手にオブジェクトだけを入れ替えたりしたため、結合テストの業務プログラムがどのような構成になっているか把握できていなかった

◆対策

結合テスト環境を運営する専任者を配置し、履歴、世代、構成管理ルールを改善。責任者の承認を得てテスト環境を提供するように改善した

◆対症療法

・構成管理ルールの策定と実行
・構成管理者の配置

◆再発防止策（教訓）

・構成管理ルールの策定
・構成管理者の配置
・構成管理ツールの決定

モチベーション（拡張）

59 自社のプロジェクト・マネージャが体調不良で倒れた

自社のプロジェクト・マネージャが体調不良で倒れてしまった

◆原因

長時間残業、責任の集中、精神的疲労などプロジェクト・マネージャは常のストレスとの戦いにあり、それが原因で体調不良に陥った

◆対策

プロジェクト・マネージャの交代はプロジェクト全体に不信感を広げるので早期の対応が必要である。リーダーの中から暫定プロジェクト・マネージャを任命する方法もあるが、失敗するとこのプロジェクト・マネージャも離脱して将棋倒しのような現象が起こり得る。プロジェクト責任者が暫定的にプロジェクト・マネージャを兼任するのがよい

◆対症療法

・新しいプロジェクト・マネージャの投入（上位者、内部）

◆再発防止策（教訓）

・上位者によるプロジェクト・マネージャへのケア
・会社の勤労管理の整備

顧客（拡張）

60 顧客側の担当者が能力・経験不足でまったく役に立たない

顧客側の担当者とまともなコミュニケーションをとれず、何を言っても十分に対応してもらえない。プロジェクトも停滞状態に陥った

◆原因

顧客側の担当者のコミュニケーション能力がなかった。システムも業務も理解していなかった

◆対策

システムを開発する上で重要な情報を顧客側の担当者から得られない場合、それに起因するリスクを一覧にまとめ、顧客側の責任者と相談する。プロジェクトとしてのリスクを客観的かつ誠実に説明して改善を求める

◆対症療法

・リスク一覧の作成
・リスクを顧客に説明する
・第三者による顧客窓口の変更依頼

◆再発防止策（教訓）

・コミュニケーション・ルールの計画

コスト

61 テスト工数を小さく見積もってしまった

パッケージ・ソフトを導入するので、実際の開発工数は少ないと思い、機能工数の積み上げで見積もってしまった

◆原因

パッケージ・ソフトの導入は、開発工数を抑えられるが、妥当性検証やインタフェース・テストなどのテスト工数がかかる。それを見積もっていなかった

◆対策

・有識者を加え、不足するテストを洗い出し、必要な期間、要員、テスト環境などのリソース投入を見積もる
・実現可能なテスト計画を再作成し、ステークホルダーに変更承認を得る
・プロジェクトの目標を達成でない場合には、品質、コスト、納期の見直しも含め、ステークホルダー間で調整する

◆対症療法

・残タスクの洗い出し
・再スケジュール
・顧客との費用の交渉
・要員投入

◆再発防止策（教訓）

・見積もりの妥当性検証
・パッケージの調査

コミュニケーション

62 開発チームへの指示系統が複数

**開発チーム（2次請け会社）への指示系統が2つになっており、現場が混乱した。
作業遅れと作業品質の低下を引き起こしていた**

◆原因

元請け会社が2次請け会社に開発を委託して、さらにその下に孫受けの開発ベンダーがいた。元請け会社が孫請けの開発ベンダーに直接指示を出していたため混乱が生じた。元請け会社の指示は、現場レベルの作業手順に落とし込まれていなかった

◆対策

元請け、2次請け、孫請けの3社による確認の場を設け、「階層を飛び越した指示は出さない、受けない」を改めて合意事項として確認する

◆対症療法

・指示系統の明確化と徹底

◆再発防止策（教訓）

・コミュニケーション・ルールの計画
・ステークホルダー俯瞰図の作成と合意

人的資源

63 個別チームの状況を把握していないため、総合テストに入っていいのか分からない

**結合テストから総合テストに進もうとする段階で、
「総合テストに入ってよいものかどうか判断できない」と悩んだ**

◆原因

プロジェクト全体の統括機能が弱く、開発重視の体制になっていたため、キーパーソンがサブシステムの開発リーダーに回されていた。これにより、総合テストに進むべきか判断できる人材がテスト・チームにいなかった

◆対策

・チームごとに「品質評価報告書」（結合テスト結果の総括）を提出させる。これにより、そのチーム（機能、サブシステム）が総合テストにいつ入れるか判断する
・その結果を基に、必要があれば結合テストの組立て（テスト順序など）を見直す。本来のテスト計画通りに結合テストができない可能性があるため
・総合テストに進むチームのリーダーに全体統括の機能を担当してもらう

◆対症療法

・品質状況の把握
・総合テスト計画の策定
・総合テストのための体制作り

◆再発防止策（教訓）

・工程区切りの評価
・品質評価基準の策定
・品質報告の徹底とレビュー
・総合テストの品質管理ルールの策定
・体制計画

人的資源

64 キーパーソンが疲弊している

危機的状況にあるプロジェクトで、プロジェクト推進のために作業がキーパーソンに集中した。キーパーソンは疲弊しきってしまい、作業効率が上がらず、さらにプロジェクトが進まなくなってしまった。それが原因で、さらにキーパーソンに負荷が増えるという悪循環が発生した

◆原因

どうしても負荷がキーパーソンに集中してしまう一方で、キーパーソン以外の人材が育っていなかった。キーパーソンが倒れたらプロジェクトは破綻してしまう

◆対策

- キーパーソンのタスクを整理する要員を付けるとともに、キーパーソンの「キー」である部分を代行できる人員を育成する
- タスク整理要員の主な任務は、集中するタスクの交通整理と重要度/期限の管理。キーパーソンは与えられたタスクを順にこなすだけの作業に専念できる
- キーパーソン代行要員の主な任務は、対象となる「キー」部分を習得し、少なくとも問題の1次切り分けと解決に必要な情報を収集できること。キーパーソンはある程度整理された情報の中から答えを出していく作業に集中できる

◆対症療法

- キーパーソンの負荷分散

◆再発防止策（教訓）

- キーパーソンに対する労務管理
- キーパーソンに負荷がかからないような作業分担
- 後継者の育成

人的資源

65 協力会社が多階層になり責任範囲があいまいに

規模が非常に大きいプロジェクトで、協力会社から納品されるプログラムやシステムの品質が悪かった。協力会社間の責任範囲があいまいで、解決のための方向性が見えない

◆原因

元請け会社の下に、2次請け会社、孫請け会社と多階層になり、各社の責任範囲が不明確になってしまった。これに起因して、品質の低下、状況把握の悪さが表面化。プロジェクトの規模に対して、統合管理チームが手薄だった

◆対策

- 中間層の協力会社が機能を果たしていないことが多いので、その場合は例外的に、会社・階層を問わない指示体制を確立する
- 協力会社が一つの場所に集まっていれば、上記の対策を進めやすい
- 中間の協力会社が孫請けの開発状況を把握できないのなら、中間の協力会社から実質的に指揮権を取り上げる（責任は持ってもらうが、管理できていない点を主張する）
- 協力会社が場所的に離れている場合は、上位の協力会社とのコミュニケーションをよくする。最後は緊急集合

◆対症療法

- 作業者を集める
- 指示・命令系統の整理
- 統合管理チームの構築

◆再発防止策（教訓）

- 協力会社（中間層）に対する管理徹底

人的資源

66 実質的なマネジメントを協力会社の開発グループが実施

元請け会社が2次請け会社に発注しているが、2次請け会社の製品がシステムの主要部分を構成しているために、実質的なプロジェクト・マネジメントを2次請け会社が担当した。その負荷が重く、肝心の開発作業が遅れてしまった

◆原因

2次請け会社がプロジェクト・マネジメントと開発作業を一緒に進めたことで、オーバーフローしてしまった

◆対策

契約時の作業分担を明確にする必要がある。特に、メーカーは2次請けの協力会社にシステム開発を丸投げするケースが増えている。メーカーは「プロジェクト・マネジメント担当だが何もしない」、協力会社は「すべてを請け負い、開発の責任もすべて持つ」、顧客は「困ったら問題を協力会社に押し付ける」という関係はシステム開発を台なしにしてしまう

◆対症療法

・作業分担の明確化
・作業の再配分

◆再発防止策（教訓）

・作業分担表の作成
・会議体の設定

人的資源

67 総合テストの進捗が悪い

上流工程の進捗はそこそこ良かったが、下流工程で総合テストが遅れだした

◆原因

総合テストをあまりスキルの高くないエンジニアが実施していた

◆対策

・要員の交替
・テスト作業の均質化を図るため、テスト・ツールや標準化されたテスト手法を採用
・テスト仕様書を見直す。スキルの低い要員でも仕様書通りに作業すればよいという環境を作る

◆対症療法

・テスト要員投入
・ルールの策定
・ツールの標準化
・教育
・スキルに応じた役割・作業分担

◆再発防止策（教訓）

・テスト計画の立案
・総合テストには顧客に参画を依頼する
・ハイスキルな要員を総合テストにあてる

人的資源

68 テスト作業に無駄が多い

テスト用マシンの使用時間の半分以上が作業手順の誤りや、テストデータの誤りによる再テストや修復という無駄な作業に費やされていた

◆原因

テスト用マシンのシステム設定情報が誤っていた。テスト用データベース/テスト用データといったテスト環境の品質が悪かった。さらに、テスト手順や障害発生時の情報取得・解析手順、復旧手順、構成管理といったテスト作業にも品質不良があった

◆対策

- テスト環境数を増やす。テストごとに設定情報の変更やデータベース/テストデータの入れ替えをしなくて済むようにする。あるいは、その頻度を少なくすることにより、テスト作業の品質不良による影響を少なくする
- 複数のテスト環境の統括管理責任者を置き、次のことを管理させる
  - テスト環境の利用計画作成/スケジュール管理
  - テスト環境の切り替え計画作成/スケジュール管理
  - 設定情報を含めたテスト環境の構成管理
  - テスト環境切り替え手順書作成と周知徹底
- 障害管理責任者を置き、次のことを管理させる。
  - 障害対応フローおよび手順書の作成と周知徹底（障害登録、チェックアウト、チェックイン、プログラム修正・テスト、テスト環境、プログラム・リリース、報告までの一連のフローを含む）。また、テスト実施者、プログラム修正者、ライブラリ管理者、障害管理者がそれぞれ何をすべきかを明確にする
  - 修正プログラムのリリース・スケジュール作成と実績のフォロー
- テスト実施時の作業フロー、手順書の作成と周知徹底
- 上記を実施することによる工程遅延と全体工程内での挽回策を顧客に説明し、承認を得る

◆対症療法

- テスト環境の整備
- テスト環境の統括責任者配置
- 障害管理責任者の配置
- テストの作業フロー・手順書の作成と徹底
- 顧客への挽回策の説明

◆再発防止策（教訓）

- テスト計画の立案と有識者レビュー（テストのプロ、業務有識者、顧客）

人的資源

69 サービス開始の可否が判断できない

進捗が大幅に遅れており、サービス開始の可否が判断できない

◆原因

本社の情報システム部門が、支社の情報システム部門に仕様策定を丸投げしていたために、本社の情報システム部門が仕様や進捗状況を把握していなかった

◆対策

・プロジェクト・マネージャ、キーパーソンのヒアリング、プロジェクト進捗状況の実情把握、問題摘出
・サービス開始時期の到達予測
・プロジェクトの再構築、再開による到達度を予測
・上記とほぼ並行して、顧客の最低必要サービス項目の抽出、スケジュール再設定
・サービス項目、サービス開始時期を再設定し、顧客、プロジェクト・マネージャ、メンバーによる打ち合わせ体制を見直す

◆対症療法

・現状把握
・プロジェクト・マネージャ支援要員の投入
・再スケジュール
・体制見直し
・リリース機能の顧客との合意

◆再発防止策（教訓）

・プロジェクト計画の策定
・プロジェクト監査の仕組みづくり

人的資源

70 総合線表がない

個々の開発チームの線表はあるが、全体を俯瞰する線表がない。
そのため、システム全体としての進捗はどうなっているのか把握できない

◆原因

プロジェクト・マネジメント能力の不足

◆対策

・プロジェクト・マネージャの交代あるいはプロジェクト・マネージャ支援者の配置
・システム統合チームの設置
・顧客のサービス必要時期の再確認
・システム統合への移行スケジュール、体制、作業項目（WBS）の策定とその実施

◆対症療法

・プロジェクト・マネージャ支援要員の投入
・プロジェクト・マネージャの交代
・顧客との必要機能の調整
・再スケジュール
・体制の見直し

◆再発防止策（教訓）

・プロジェクト計画の策定
・プロジェクト監査の仕組みづくり
・工程区切り監査

人的資源

71 顧客側と開発側のプロジェクトの一体感欠如

顧客側と開発側の間にプロジェクトとしての一体感がない

◆原因

開発側のプロジェクト体制が手薄

◆対策

・プロジェクト推進会議を設置する。顧客側、開発側双方のキーパーソンと有識者による会議体を設定し、重要課題の検討など、プロジェクトの方向付けや助言を行う
・開発側体制を見直す。業務知識を持ったメンバーの投入やPMOメンバーによるマネジメント支援を要請する
・段階的に機能をリリースするなど、スケジュールを見直す。スケジュールの見直しは顧客側キーパーソンを含めて検討し、同意を得るとともに一体となって対応する意識を高める
・進捗会議には顧客側キーパーソンの参加を求め、情報共有を促進する

◆対症療法

・会議体の策定による一体感の醸成
・プロジェクト・マネージャ支援要員の投入

◆再発防止策（教訓）

・チームビルディング
・コミュニケーション・ルールの計画
・非公式なコミュニケーション手段の形成

スコープ

72 オープンシステムのプロジェクト経験がない

オープン系システムの構築経験がなかったため、システム品質の低下や、進捗遅延を招いた

◆原因

オープン系システムの構築、特にマルチベンダー/マルチプロダクトによるオープン系分散処理システムには、多数の製品ベンダーが絡むため、高度な技術力とマネジメント力が要求される。しかし、統括部門に経験がなく、要員もいなかったために、取りまとめの能力がなかった

◆対策

・至急、オープン系システムの経験者、専門家をプロジェクトに動員できるようにする
・オープン系製品群を統合利用したことがある経験者の助言を受けられるようにする
・自社内に人材が不足していれば、社外に人材を求める
・使用する個別製品の経験者を確保する。製品ベンダー（開発元、販売元）のサポート部門との関係を構築する。あるいは、開発チームをオープン系システムの経験者に変更する

◆対症療法

・有識者の投入
・製品ベンダーとのサポート体制を構築

◆再発防止策（教訓）

・リスク管理の強化
・チームビルディング

スコープ

73 要求仕様が固まらないまま開発に突入

要求仕様が固まらず、要求仕様書がないまま開発が進み、プロジェクトが破綻した

◆原因

プロジェクト・マネジメント能力の不足。最近の小規模プロジェクトでは、このように要求仕様書が確定しないまま開発に入るのは当たり前になっている。後付けで要求仕様書を作成するケースが多い

◆対策

- 本来、要求仕様書を作らずに開発を進めるべきではない。開発を一度止めて、改めて要件定義を確定させるタスクを設ける。その時点までの設計書を活用して、できるだけ期間短縮を図る
- プロトタイピングを実施して主要機能の画面、帳票などを簡易に作成し、プロトタイプで実質的に要求仕様を確認する
- 開発規模が小さい場合（7人以内ぐらい）は、XP（Extreme Programming）手法に変更して対応する

◆対症療法

- 開発を一度止める
- 要求仕様の確定とドキュメント化
- プロトタイプの作成

◆再発防止策（教訓）

- 開発プロセスの構築（開発計画書に記載）
- 顧客に開発プロセスを合意してもらう

タイム

74 プロジェクト全体に疲弊感が漂っている

プロジェクト全体に疲弊感が漂っており、現場の様子を見ていてもモチベーションの低下が見られる

◆原因

疲弊感やモチベーション低下の原因はさまざまだが、出口の見えないプロジェクトや、先の見えないテストに明け暮れる日々がモチベーションを低下させた

◆対策

- チーム・メンバーとのコミュニケーションを通じて疲弊感やモチベーション低下の原因を探る
- プロジェクトの目標と現状を再確認し、目的達成に向け、作業の優先順位付けや最適化を行い、実行可能な計画に見直す
- 再度、クリティカルパスやマイルストーンをメンバーで共通認識する（ゴールを再確認する）

◆対症療法

- プロジェクトの目標の再確認
- チーム・メンバーとのコミュニケーション
- 作業の優先順位付け
- 計画の見直し
- ゴールの設定

◆再発防止策（教訓）

- 労務管理の徹底
- チームビルディング

調達

75 外部調達のパッケージ製品の品質が悪くて使えない

外部から調達したパッケージ製品の品質が悪い

◆原因

外部からのパッケージ製品の調達に関しては、そのパッケージがどのようなサービス・レベルをターゲットに作られているのかを事前に検証しておく必要がある。この事例では、ミッションクリティカル性を想定していないパッケージ製品だったことにより問題が生じた

◆対策

- パッケージ製品利用時には、パッケージ製品をサポートする会社と必ずサポート契約を結び、逃げられない体制を作る
- 海外の会社に対しては、製品使用の制限事項や用途の制限について事前に調査する
- パッケージを使うには、パッケージに合うよう顧客のビジネス・プロセスを変更する必要があることを十分顧客に認識してもらう

◆対症療法

- サポート契約の締結
- パッケージの調査
- 顧客との情報共有

◆再発防止策（教訓）

- 事前調査（サービスレベル）
- パッケージ・ベンダーとの契約事項の確認

品質

76 プログラムの品質が悪い

プログラム品質が悪く、テストを実施してもバグだらけだった

◆原因

プログラミング能力の不足やレビューの不足

◆対策

- プログラマごとに少なくとも１本のソースコード・レビューをして、スキル不足のプログラマを識別する。さらに、そのプログラムの問題点を抽出し、修正方針、方法を決める。レビューの負担が重い場合は、第三者を投入してレビューする。スタイルチェックは静的ソースコード解析ツールを使う
- スキル不足のプログラマが作成したプログラムをすべて修正する。修正作業は、同一チーム内のスキルのあるプログラマが分担する
- スキル不足のプログラマをプログラミングから外し、その後の障害対応のためのプログラム修正も同一チーム内のスキルのあるプログラマに分担してもらう。外したプログラマには、可能であれば、修正プログラムのリリース前修正確認、リグレッション・テストを実施してもらう
- ソースコード・レビュー後の修正プログラムについて結合テストを再度実施し、不具合を修正して、品質を上げる
- 上記を実施することによる工程遅延と全体工程内での挽回策を顧客に説明し、承認を得る

◆対症療法

- コードレビュー、ピア（peer）レビューの実施
- プログラム修正とテストの再実施
- スキル不足のプログラマの識別

◆再発防止策（教訓）

- プログラマのスキルチェック
- レビューの励行
- ドキュメントの記述方法の充実
- コーディング規約の策定と徹底

品質

77 パフォーマンス設計の不備

レスポンスの確保や無応答状態をなくすための方式設計を全くしておらず、トラフィック量やトラフィック・パターンも把握していなかった。このため、総合テストに入るころに性能問題が表面化した

◆原因

多くの技術者が、「性能上の問題が起こったらハードを増やせばいい」と安易に考えていた。しかし、性能問題は設計時点で考慮すべきだった

◆対策

・性能問題の状況把握
・多くの場合、パフォーマンス・チューニングの専門家を投入
・性能支援チームの構築
・当初の性能モデル、トラフィック条件と現時点での差異を顧客とともに確認
・システムに求められる性能要件の再設定
・性能ボトルネックの検出と改善
・性能チューニング・性能確認テスト

◆対症療法

・性能問題の専門化の投入
・性能要件の確認
・ボトルネックの抽出と分析
・対処方法の検討と実施
・顧客との性能要件の合意

◆再発防止策（教訓）

・有識者による方式設計レビュー
・顧客への性能要件の確認

付録 4.測定項目リスト

知識エリア

重要度	No	測定可能な概念/測定目的	測定データ項目
		◆測定データの属性（参考） 具体的な測定項目の例、属性 ◆測定方法 現物確認▶測定項目のソースとなる文書 システマティック確認▶ツールによる測定方法	◆測定データの見方、分かること 測定データが示す「意味」の説明。カッコ内の記号は、ヒアリングシートのチェック項目番号に対応

スコープ

重要度	No	測定可能な概念/測定目的	測定データ項目
★	1	機能的規模と変動（機能面でスコープが増加していないか）	機能変更要求数（未、済）
		◆測定データの属性（参考） 機能ごと、要件ごとの変更要求数 ◆測定方法 現物確認▶変更管理表	◆測定データの見方、分かること 機能変更要求数が増加傾向にある場合、 ・機能凍結時期が明確になっておらず、追加要求に歯止めがかかっていない可能性がある（H3） ・基本設計書でのスコープが明確でない。あるいは顧客の承認がとれていない可能性がある（H3） ・機能変更対応する場合の費用について顧客との合意がとれていない可能性がある（H11,H12） ・顧客の機能変更要求マナーの悪さが分かる

スコープ

重要度	No	測定可能な概念/測定目的	測定データ項目
★	2	機能的規模と変動（機能面でスコープが増加していないか）	機能変更対応数（仕様変更対応数）
		◆測定データの属性（参考） 機能ごと、要件ごとの変更要求数 ◆測定方法 現物確認▶変更管理表	◆測定データの見方、分かること 機能変更対応数・対応予定数が増加傾向にある場合、 ・機能変更対応する場合の費用について顧客との合意がとれていない可能性がある（H3,H11,H12） ・プログラム変更時のレビューが不十分の可能性がある（H39） ・今後対応するのに必要な期間、負荷が分かる ・ドキュメント変更が不十分な場合がある

スコープ

重要度	No	測定可能な概念/測定目的	測定データ項目
	3	機能的規模と変動（機能面でスコープが増加していないか）	ドキュメントの規模（ページ数）
		◆測定データの属性（参考） ドキュメントの数の推移、ドキュメントごとのページ数の推移 ◆測定方法 現物確認▶プロジェクト実行計画書、テスト計画書、基本設計書、詳細設計書 システマティック確認▶構成管理ツール（EPMツール）	◆測定データの見方、分かること ・ドキュメント数、ページ数が増加している場合、機能追加されている可能性がある（H3） ・機能変更対応数と比較して、ドキュメント数、ページ数が増加していなければ、機能変更がドキュメントに反映されていない可能性がある ・そのドキュメントを基に作られたテストケースでテストをしている場合、テストが十分でない可能性がある

スコープ

★ **4** 機能的規模と変動（機能面でスコープが増加していないか）

ソースコード行数

◆測定データの属性（参考）

ソースコード行数の推移

◆測定方法

システマティック確認▶
構成管理ツール（EPMツール）

◆測定データの見方、分かること

ソースコード行数が増加傾向にある場合、
・機能変更対応が多く行われている可能性がある。この場合、顧客の追加要求に歯止めがかかっていない可能性がある（H3）
・機能変更対応に起因したプログラムのデグレードが発生している可能性が高い

コード行数が大きく減っている場合、誤って削除した可能性が高い（不要コードを削除して整理したなどの理由があればよい）

スコープ

5 機能的規模と変動（機能面でスコープが増加していないか）

ソースコードの変更行数

◆測定データの属性（参考）

累積ソースコード変更行数

◆測定方法

システマティック確認▶
構成管理ツール（EPMツール）

◆測定データの見方、分かること

・ソースコード変更行数が増加している場合、
・機能変更対応が多く行われている可能性がある。この場合、顧客の変更要求に歯止めがかかっていない可能性がある（H3）
・機能変更対応に起因したプログラムのデグレードが発生している可能性が高い
・バグが多発して、その対応をしている可能性がある。この場合、バグ発生件数の推移と比較して、プログラム改修のタイミングが妥当であるか分かる

タイム

★ **6** マイルストーンの達成状況の管理

結合テスト工程での作業単位の進捗

◆測定データの属性（参考）

責任者
マイルストーンの成果物検証数（未、済）
成果物（未）の進捗率（進捗の程度）

◆測定方法

現物確認▶ガントチャート、稲妻線

◆測定データの見方、分かること

・マイルストーンの対象成果物の検証数と、成果物に対する達成度により、マイルストーン達成の程度が分かる（達成の度合いは、成果物の性質に応じて適切に定義されていること）
・マイルストーン達成責任者が明確でないと、達成のための種々の管理ができていない可能性がある

タイム

★ **7** マイルストーンの達成状況の管理

基本設計～プログラミング・単体テストまでの進捗

◆測定データの属性（参考）

責任者
マイルストーンの成果物検証数（未、済）
成果物（未）の進捗率（進捗の程度）

◆測定方法

現物確認▶ガントチャート、稲妻線

◆測定データの見方、分かること

・マイルストーンの対象物成果物の検証数と、成果物に対する達成度により、マイルストーン達成の程度が分かる（達成の度合いは、成果物の性質に応じて適切に定義されていること）（H3）
・マイルストーン達成責任者が明確でないと、達成のための種々の管理ができていない可能性がある

タイム

8 マススケジュールの妥当性 — マススケジュールのオーバーラップ度

◆測定データの属性（参考）

◆測定方法

現物確認▶ガントチャート、稲妻線

◆測定データの見方、分かること

・マススケジュールの作業項目で同時に進行している数が多い場合、スケジュールの整合性がとれていない可能性がある（H19）
・スケジュールと要員山積みの整合性がとれていない可能性がある（H19）
・オーバーラップしているフェーズ間の成果物（機能）の整合性が保たれていない可能性がある

タイム

★

9 クリティカルパススケジュールの達成状況の管理 — 結合テスト工程でのクリティカルパス作業の進捗

◆測定データの属性（参考）

進捗率
作業担当者

◆測定方法

現物確認▶WBS、ガントチャート、稲妻線
システマティック確認▶EVM

◆測定データの見方、分かること

・進捗率、ガントチャートにより、クリティカルパスのスケジュール遅延の程度が分かる（H23）
・作業担当者が明確でないと、他の作業との優先度が不明確になりスケジュール遅延の可能性がある

タイム

10 重要な機能の達成状況の管理 — 重要な機能数

◆測定データの属性（参考）

◆測定方法

現物確認▶テスト計画書

◆測定データの見方、分かること

重要な機能（進捗が遅れるとクリティカルパスになる機能）の規模感が分かる。
重要な機能はプロジェクト開始時に定めておくことが前提（H2）

タイム

★

11 重要な機能の達成状況の管理 — 重要な機能の作業進捗

◆測定データの属性（参考）

作業ごとの進捗率
作業ごとの担当者

◆測定方法

現物確認▶WBS、ガントチャート、稲妻線

◆測定データの見方、分かること

・進捗率、ガントチャートにより、スケジュール遅延の程度が分かる
・進捗率から残作業量の程度が分かる
・重要な機能の作業が遅れると、この作業がクリティカルパスになる可能性が高い（H2）
・作業者が明確でないと、他の作業との優先度が不明確になり、スケジュール遅延の可能性がある（H17）

タイム

★ **12** 作業進捗（テストの進捗）

テストケース消化数

◆測定データの属性（参考）

日ごとのテストケース完了数、残数、合格数、不合格数

◆測定方法

現物確認▶テストケース表

◆測定データの見方、分かること

・テストケース実施数、合格数の計画と実績の推移より進捗が分かる（H22）
・実施数、不合格数、合格数の推移により、テストが終了する時期が分かる

タイム

★ **13** 作業進捗（テストの進捗）

テストチームごとのテストケース消化数

◆測定データの属性（参考）

日ごとのテストケース完了数、残数、合格数、不合格数、テスト実施者一人当たりのテストケース消化数

◆測定方法

現物確認▶テストケース表、テスト体制表

◆測定データの見方、分かること

・テストケース実施数、合格数の推移よりチームのテスト作業進捗が分かる（H22）
・テストケース実施数、合格数が予定より大幅に遅れている場合、そのチームのテスト作業要員の数あるいは力量不足が分かる（H21）
・テスト実施者一人当たりのテストケース消化数が平均より大幅に低い場合、テスト作業者の力量不足の可能性がある（H21）

タイム

★ **14** 作業進捗（プログラム修正の進捗）

チェックイン/アウト回数の推移（ソースコード更新頻度の推移）

◆測定データの属性（参考）

チェックイン/アウトの契機、対象ファイル、日時、目的、責任者（実施者）

◆測定方法

現物確認▶プログラム変更管理表
システマティック確認▶構成管理ツール、EPMツール

◆測定データの見方、分かること

・検出バグ（現象）数の推移と比較して、チェックイン/アウトの頻度が少ない場合、構成管理、リリース管理がなされていない可能性がある（H6）。あるいはプログラム改修対応力不足の可能性があり、進捗遅れの可能性がある
・特定のプログラムのチェックイン/チェックアウトの頻度が異常に多い場合、そのプログラムの品質が悪い場合がある

タイム

15 作業進捗（プログラム修正の進捗）

ソースコード量の推移

◆測定データの属性（参考）

プログラム変更件数

◆測定方法

現物確認▶プログラム変更管理表
システマティック確認▶構成管理ツール

◆測定データの見方、分かること

・機能変更、バグ修正にともなうプログラム修正の進捗の程度が分かる
・検出バグ（原因）数と比べてソースコード量の増加があまりにも少ない場合、バグ修正が不十分である可能性がある
・コード行数が大きく減っている場合、誤って削除した可能性が高い（不要コードを削除して整理したなどの理由があればよい）

タイム

16 作業進捗（プログラム修正の進捗）

ソースコード改変量の推移

◆測定データの属性（参考）

プログラム変更件数

◆測定方法

現物確認▶プログラム変更管理表
システマティック確認▶構成管理ツール

◆測定データの見方、分かること

・機能変更、バグ修正にともなうプログラム修正の進捗の程度が分かる
・検出バグ（原因）数と比べてソースコード改変量があまりにも少ない場合、バグ修正が不十分である可能性がある
・検出バグ（原因）数の推移との比較で、説明できないソースコード改変量の動きがあれば、異常が発生している可能性がある

タイム

17 作業進捗（ドキュメント修正の進捗）

ドキュメント量の推移

◆測定データの属性（参考）

ドキュメントの数の推移、ドキュメントごとのページ数の推移

◆測定方法

現物確認▶ドキュメント変更管理表
システマティック確認▶構成管理ツール

◆測定データの見方、分かること

機能変更およびテストケース見直しにともなうドキュメントの追加、修正の進捗の程度が分かる

タイム

18 作業進捗（ドキュメント修正の進捗）

ドキュメント改編量の推移

◆測定データの属性（参考）

修正したドキュメントの数の推移、
ドキュメントごとの改変したページ数の推移

◆測定方法

現物確認▶ドキュメント変更管理表
システマティック確認▶構成管理ツール

◆測定データの見方、分かること

・機能変更およびテストケースの見直しにともなうドキュメントの追加、修正の進捗の程度が分かる

タイム

★

19 開発・テスト環境の充足度

開発環境の数

◆測定データの属性（参考）

◆測定方法

現物確認▶開発環境構成図

◆測定データの見方、分かること

開発環境数が少ないと、プログラム改修のための開発環境の調整や設定変更に時間がかかり、改修速度を遅らせている可能性がある

タイム

★ **20** 開発・テスト環境の充足度

テスト環境の数

◆測定データの属性（参考）

◆測定データの見方、分かること

テスト環境数が少ないと、改修したプログラムのテストのための環境の調整や設定変更に時間がかかり、改修速度を遅らせている可能性がある

◆測定方法

現物確認▶テスト計画書、テスト環境構成図

コスト

★ **21** プロジェクト予算の管理状況

使った金額

◆測定データの属性（参考）

追加工数、金額

◆測定データの見方、分かること

計画と実績の差（追加工数、金額）が大きい場合、計画時の見積もり前提が不明確、あるいは見積もりそのものが甘い可能性が高い。その場合、変更対応の工数を顧客に請求できていない量が多い可能性がある（H3）

◆測定方法

現物確認▶プロジェクト収益管理表

コスト

★ **22** プロジェクト予算の管理状況

アーンド・バリュー

◆測定データの属性（参考）

◆測定データの見方、分かること

・スケジュール遅れ、コスト超過の大きいタスク、チームを抽出できる
・スケジュールやコストの改善、悪化の傾向が分かる。今後の到達点を予測できる

◆測定方法

システマティック確認▶EVMツール

コスト

★ **23** 機能追加時のコスト面での対応状況

追加請求額

◆測定データの属性（参考）

◆測定データの見方、分かること

機能変更対応数と比べて顧客への追加請求が少ない場合、スコープ変更時の対応方法について顧客と取り決めができていない可能性がある（H12、H15、H24）

◆測定方法

現物確認▶追加請求書

コスト

24 プロジェクト予算の余裕度 — 残っている予算

◆測定データの属性（参考）

◆測定方法

現物確認▶プロジェクト収益管理表

◆測定データの見方、分かること

・今後の対応に必要な費用が、残っている予算を超過している場合、計画時の見積もりが甘い可能性がある（H3）
・超過金額に応じて、今後の対応を実施すべきか、上層部の了解を取る必要があるかどうかの判断材料になる（H12）

品質

★ 25 信頼性 — 検出バグ（現象）数

◆測定データの属性（参考）

発生日、重要度、修正完了数、修正残数、欠陥密度（検出バグ（現象）数/生産物量）

◆測定方法

現物確認▶バグ表、プログラム変更票、（プログラム変更管理リスト）、品質評価報告書、バグ分析表
システマティック確認▶EPMツール

◆測定データの見方、分かること

・検出バグ数累計、欠陥密度、重要度別バグ発生比率からプログラム品質の程度が分かる
・発生推移から飽和数、飽和時期の概要が分かる
・バグ検出数の推移と修正完了数の推移から全数修正完了時期が見込める（H22）
・バグ修正完了数の推移からバグ修正の対応力不足、次工程のスケジュールへの影響が分かる
・バグ発生数に比べてプログラム修正数が異常に少ない場合、プログラム変更管理がされていない、あるいはプログラム変更内容のレビューがされていない可能性がある（H31、H39）

品質

★ 26 信頼性 — 開発チームごとの検出バグ（現象）数

◆測定データの属性（参考）

発生日、重要度、修正完了数、修正残数

◆測定方法

現物確認▶バグ票、品質評価報告書

◆測定データの見方、分かること

・バグ発生数の多い開発チームには、品質基準が明確に伝わっていない可能性がある。あるいは、担当する機能の技術的難易度が高いか、仕様の確定度が低い可能性がある（H57）
・開発チームごとの品質への取り組み姿勢が分かる

品質

★ 27 信頼性 — MTBF（平均故障間隔）

◆測定データの属性（参考）

開発チームごと、機能ごと、プログラムごと

◆測定方法

現物確認▶バグ表（発見日、修正日）
システマティック確認▶EPMツール

◆測定データの見方、分かること

クリティカルな機能のMTBFが短いと品質が悪い（触れば直ぐ不良が発生する）

品質

★ **28** 信頼性

MTTR（平均修正日数）

◆測定データの属性（参考）

開発チームごと、機能ごと、プログラムごと

◆測定方法

現物確認▶バグ票（発見日、修正日）
システマティック確認▶EPMツール

◆測定データの見方、分かること

MTTRが長い場合、
・プログラム構造が悪く、修正に時間がかかっている可能性がある。設計が悪い、あるいは設計されていない可能性が高い
・修正に関して技術的に難しい問題がある、あるいは開発チームの対応力不足の可能性がある

品質

29 信頼性
（ドキュメントは修正されているか）

修正内容のドキュメントへの反映件数

◆測定データの属性（参考）

バグ修正にともないドキュメントを修正した件数

◆測定方法

現物確認▶バグ票、ドキュメント変更管理表

◆測定データの見方、分かること

検出バグ（現象）数と比べてドキュメント変更数があまりにも少ない場合、バグ修正内容がドキュメントに反映されていない可能性がある（H6）

品質

30 信頼性

類似バグ調査の実施回数

◆測定データの属性（参考）

◆測定方法

現物確認▶バグ票、バグ分析表

◆測定データの見方、分かること

類似バグ調査が実施されていない、あるいはその回数が少ない場合、バグ修正が不十分で、類似のバグが発生する可能性が高い（H31）

品質

★ **31** 信頼性

検出バグ数（現象）の分析実施回数

◆測定データの属性（参考）

◆測定方法

現物確認▶バグ票、バグ分析表

◆測定データの見方、分かること

バグ分析が実施されていない、あるいはその回数が少ない場合、バグの除去が行われていないか、不適切である（根本的な対策が打てていない）可能性が高い（H31）

品質

★ 32 信頼性（テスト計画書はレビューされているか

テスト計画書のレビュー回数

◆測定データの属性（参考）

レビュー実施回数（計画、実績）

◆測定方法

現物確認▶テスト計画書のレビュー記録

◆測定データの見方、分かること

テスト計画書のレビューがされていない、あるいはその回数が少ない場合、テストの網羅性が低い可能性がある（H33）

品質

★ 33 信頼性（テスト計画書はレビューされているか）

テスト計画のレビュー時指摘数

◆測定データの属性（参考）

指摘数、指摘修正数

◆測定方法

現物確認▶テスト計画書のレビュー記録、指摘項目リスト

◆測定データの見方、分かること

・レビュー時の指摘数が少ない場合、レビュー不足の可能性がある（H33）
・レビュー時の指摘数が多い、あるいは修正残数が多い場合、テスト計画が不十分な可能性がある（H33）

品質

★ 34 信頼性（テストケースはレビューされているか）

テストケースのレビュー回数

◆測定データの属性（参考）

レビュー実施回数（計画、実績）

◆測定方法

現物確認▶テストケースのレビュー記録

◆測定データの見方、分かること

・テストケースのレビューがされていない、あるいはその回数が少ない場合、テストの網羅性が不十分の可能性がある。また、異常系のテストケース漏れの可能性がある（H27）
・テストケースのレビューが実施されていないのに、テストケースの作成が完了している場合、進捗率の値が感覚的に報告されている可能性がある（H22）

品質

★ 35 信頼性（テストケースはレビューされているか）

テストケース・レビュー時の指摘数

◆測定データの属性（参考）

指摘数、指摘修正数

◆測定方法

現物確認▶テストケースのレビュー記録、指摘項目リスト

◆測定データの見方、分かること

・レビュー時の指摘数が少ない場合、レビュー不足の可能性がある（H27）
・レビュー時の指摘数が多い、あるいは修正残数が多い場合、テストケースの検討が不十分か、重要性の認識が甘い可能性がある（H27）

品質

★ 36 信頼性（テストデータはレビューされているか）

テストデータのレビュー回数

◆測定データの属性（参考）

実施回数（計画、実績）

◆測定方法

現物確認▶テストデータのレビュー記録

◆測定データの見方、分かること

テストデータのレビューがされていない、あるいは回数が少ない場合、テストデータがテスト内容に対して妥当でない可能性がある（H27）

品質

★ 37 信頼性（テストデータはレビューされているか）

テストデータのレビュー時指摘数

◆測定データの属性（参考）

指摘数、指摘修正数

◆測定方法

現物確認▶テストデータのレビュー記録、指摘項目リスト

◆測定データの見方、分かること

・レビュー時の指摘数が少ない場合、レビュー不足の可能性がある（H27）
・レビュー時の指摘数が多いあるいは修正残数が多い場合、テストデータの検討不十分あるいは重要性の認識があまい可能性がある（H27）

品質

★ 38 信頼性（基本設計書はレビューされているか）

基本設計書のレビュー回数

◆測定データの属性（参考）

基本設計書レビュー実施回数(計画、実績)

◆測定方法

現物確認▶基本設計書のレビュー記録

◆測定データの見方、分かること

・レビューされていない、あるいはその回数が少ない場合、基本設計の品質不良により詳細設計、プログラミングの品質に問題を起こしている可能性がある
・市販ツール、パッケージ利用に対する評価検討書、システムアーキテクチャ設計書など技術面のドキュメントがレビューされていない、あるいは回数が少ない場合、品質だけでなく性能の問題を起こしている可能性がある（H67）

品質

★ 39 信頼性（基本設計書はレビューされているか）

基本設計書のレビュー時指摘数

◆測定データの属性（参考）

指摘数、指摘修正数

◆測定方法

現物確認▶基本設計書のレビュー記録、指摘項目リスト

◆測定データの見方、分かること

・レビュー時の指摘数が少ない場合、レビュー不足の可能性がある（H67）
・レビュー時の指摘数が多い、あるいは修正残数が多い場合は、基本設計が不十分なために詳細設計、プログラミングの品質、システムの性能に問題を起こしている可能性がある（H67）

品質

★ **40** 信頼性 **コードクローン**

◆測定データの属性（参考）

個人ごとのコードクローン量、コードクローン個所

◆測定方法

システマティック確認▶コードクローン分析ツール（EPMツール）

◆測定データの見方、分かること

・計画されていないコードクローン量が多い場合は、プログラムの品質が悪いことが分かる
・問題が起こったとき、コードクローンによる影響範囲が分かる

品質

41 信頼性 **データ項目の数**

◆測定データの属性（参考）

テーブルの数、項目の数の変動

◆測定方法

現物確認▶データベース自体、データベース定義書の変更履歴
システマティック確認▶DB管理ツール

◆測定データの見方、分かること

結合テスト段階で、データベース構造が変動している場合、上流の設計品質が極めて悪い可能性がある

品質

42 信頼性 **テストケース数**

◆測定データの属性（参考）

テストケース数の変動

◆測定方法

現物確認▶テストケース表

◆測定データの見方、分かること

・機能変更数の推移とのテストケース数の推移の比較で、機能変更に対応してテストケースが追加されているかが分かる
・テストケース数が増加している場合、テストケース設計が悪い可能性がある

品質

★ **43** 信頼性 **テストケース密度**

◆測定データの属性（参考）

テストケース数÷規模（ソースコード行数またはFP）

◆測定方法

現物確認▶テストケース表、品質評価報告書

◆測定データの見方、分かること

テストケース密度が、基準値と比較して低い場合、テストの網羅性が不足し、品質保証できない可能性がある

品質

★ 44 信頼性

手戻りテスト回数

◆測定データの属性（参考）

同じテスト項目を繰り返した数

◆測定方法

現物確認▶テストケース表

◆測定データの見方、分かること

手戻り回数が多い場合、
・リリースミスが多い、あるいは構成管理が不十分である可能性がある
・プログラム修正後の単体テストが不十分である可能性がある

品質

45 信頼性

プログラムごとのコーディング規約の順守性

◆測定データの属性（参考）

コーディング規約との違反数

◆測定方法

システマティック確認▶ソースの静的解析ツール

◆測定データの見方、分かること

コーディング規約違反数が多いと、
・プログラム品質が悪い可能性がある
・個人、チーム内に規約を順守する姿勢が足らない可能性がある（H28）

品質

46 信頼性

プログラムごとの単体テストケースの網羅率

◆測定データの属性（参考）

ロジックの網羅数（C0、C1カバレッジ）

◆測定方法

現物確認▶品質評価報告書
システマティック確認▶テスト・カバレッジ測定ツール

◆測定データの見方、分かること

テスト網羅性が低いと、プログラム品質が悪い可能性がある（H29）

人的資源

★ 47 体制の強弱（経験の程度）

テスト工程に入っている基本設計者の人数

◆測定データの属性（参考）

◆測定方法

現物確認▶体制表、キャリアシート

◆測定データの見方、分かること

結合テストケース設計に参画する基本設計者の工数が少ないと、テストケースの網羅性、正確性に欠け、テストにならない可能性がある

人的資源

48 体制の強弱（経験の程度）

新人比率

◆測定データの属性（参考）

◆測定データの見方、分かること

新人比率が大きいにもかかわらず、新人をサポートするメンバーがいない場合、チームのタスクの生産性・品質が悪い可能性が高い

◆測定方法

現物確認▶体制表、キャリアシート

人的資源

49 体制の強弱（経験の程度）

プロジェクト・マネージャの経験年数・経験プロジェクト数

◆測定データの属性（参考）

◆測定データの見方、分かること

プロジェクト・マネージャの経験年数、経験プロジェクト数が少ないにもかかわらず、プロジェクト・マネージャをサポートするメンバーがいない場合、テストの進捗管理・品質管理が十分になされていない可能性がある

◆測定方法

現物確認▶体制表、キャリアシート

人的資源

50 体制の強弱（経験の程度）

対象業務経験者数

◆測定データの属性（参考）

◆測定データの見方、分かること

対象業務経験者もテストケース設計を行う。その工数が少ないと、テストケースの網羅性、正確性に欠け、テストにならない可能性がある

◆測定方法

現物確認▶体制表、キャリアシート

人的資源

★ **51** 体制の強弱（経験の程度）

テスト経験者の数

◆測定データの属性（参考）

◆測定データの見方、分かること

・テスト経験者が少ないと、テストの最初の立ち上がりが遅くなる。テスト・スケジュールにそれが考慮されていない場合、スケジュールを守れない可能性がある

◆測定方法

現物確認▶体制表、キャリアシート

人的資源

52 体制の強弱（専任の程度） メンバーのプロジェクト間の兼務率

◆測定データの属性（参考）

メンバーの当該プロジェクトに投入できる負荷の割合

◆測定方法

現物確認▶体制表、キャリアシート

◆測定データの見方、分かること

・兼任メンバーが多い場合、そのチームの担当タスクに漏れがある可能性がある
・兼任メンバーが多い場合、兼務先プロジェクトの状況に影響を受け、当該プロジェクトでの業務を正常にこなせず、工程遅延や品質低下を招く恐れがある
・兼任メンバーが多いと、当該プロジェクトへの帰属意識が乏しく、モチベーションが上がらない可能性がある。特に、マトリクス組織による兼務では、職制上の上司の権限が、プロジェクト・マネージャの権限より強いため、プロジェクト・マネージャのコントロールが及ばなくなる可能性がある

人的資源

53 体制の強弱（専任の程度） メンバーのプロジェクト内兼務率

◆測定データの属性（参考）

同一担当者のタスク数、同一タスクの担当者数

◆測定方法

現物確認▶WBS

◆測定データの見方、分かること

同一担当者のタスク数が多い場合、
・タスク内容が明確でないまま担当者をアサインしている可能性がある（H40）
・タスクが明確な場合、担当者（責任者）の負荷を考えないでアサインしている可能性がある（H40）

同一タスクの担当者数が多い場合
・タスクが明確に分離されておらず、担当者の責任が不明確になっている可能性がある（H40）

上記いずれの場合も、管理者が担当者ごとの作業負荷をコントロールできなくなっている可能性がある（H17）

人的資源

54 体制の強弱（専任の程度） 品質管理担当者の（プロジェクト内）専任者数

◆測定データの属性（参考）

◆測定方法

現物確認▶体制表、キャリアシート

◆測定データの見方、分かること

品質管理の専任者が少ない、あるいは専任者がいない場合、品質管理、品質保証活動が十分に行われず、バグ発生状況の把握不十分、バグ分析不足のため、バグ修正対応が不十分な可能性がある

人的資源

55 体制の強弱（専任の程度） テスト・開発環境管理者の（プロジェクト内）専任者数

◆測定データの属性（参考）

◆測定方法

現物確認▶体制表、キャリアシート

◆測定データの見方、分かること

テスト環境管理の専任者が少ない、あるいは専任者がいない場合、
・テスト環境の利用スケジュール調整が不十分で、テスト環境の取り合いが発生し、その調整に時間を要し、テストの進捗を遅らせている可能性がある
・テスト環境の切り替えが計画通りできなくて、次のテストが実施できないために進捗を遅らせている可能性がある

開発環境管理の専任者が少ない、あるいは専任者がいない場合、
・開発環境の構成管理が不十分となり、プログラム修正ミスを起こしている可能性がある

人的資源

★ **56** 体制の強弱（専任の程度）

構成管理者（ライブラリアン）の（プロジェクト内）専任者数

◆測定データの属性（参考）

◆測定方法

現物確認▶体制表、キャリアシート

◆測定データの見方、分かること

・構成管理者が専任されていない場合、テスト中、頻繁に改修するプログラム、ドキュメントの構成管理ができていない可能性がある
・その場合、改修ミス、テスト環境へのリリースミスを起こし、テストを進めることができなくなり進捗を遅らせることになる

人的資源

★ **57** 体制の強弱（専任の程度）

リリース管理者の（プロジェクト内）専任者数

◆測定データの属性（参考）

◆測定方法

現物確認▶体制表、キャリアシート

◆測定データの見方、分かること

・リリース管理者が専任されていない場合、改修プログラムのリリース・スケジュール、リリース事由が管理されていない可能性がある
・その場合、テスト環境へのリリースミスを起こし、テストを進めることができなくなり、進捗を遅らせることになる

人的資源

58 体制の強弱（技術レベル）

ITSS（ITスキル標準）のレベル

◆測定データの属性（参考）

◆測定方法

現物確認▶体制表、キャリアシート

◆測定データの見方、分かること

・プロジェクトの規模に応じたスキル・経験を持つプロジェクト・マネージャあるいはプロジェクト・マネージャ支援要員がアサインされていない場合、テストの進捗管理・品質管理が十分になされていない可能性がある（H72）
・プロジェクトの規模に応じたスキル・経験を持つアプリケーション・スペシャリスト、ITアーキテクトなどがアサインされていない場合、品質の問題を起こしている可能性がある。また、バグ発生時修正対応の生産性が悪い可能性がある

コミュニケーション

★ **59** 会議の出席状況

会議出席率

◆測定データの属性（参考）

会議開催頻度（計画、実績）、会議出席者数（計画、実績）

◆測定方法

◆測定データの見方、分かること

会議の開催頻度、出席率が低い場合、
・コミュニーケーションが不十分である可能性がある（H45）
・担当範囲外に無関心になっている可能性がある（H80）
・自分の仕事に影響しないことには無関心となり、体制が維持できなくなる可能性がある
・出席率が低いなかで、プロジェクト全体の情報を収集してまとめる管理業務の負荷が大きくなっている可能性がある

コミュニケーション

60 メール送受状況

メール送信数

◆測定データの属性（参考）

個人ごとの送信数、メール・タイトルごとの送信数

◆測定方法

システマティック確認▶Mailman（EPMツール）

◆測定データの見方、分かること

・危機的な状況のとき、メールを発信しない人がいる場合、その人がコミュニケーションできていない可能性が高い（H81）
・個人ごと、メール・タイトルごとのメール送信数の異常値を見て、理由のつかない異常値があれば、なにかの問題がある可能性がある（H81）

コミュニケーション

61 協力会社との連携状況

チームごとの作業場所数

◆測定データの属性（参考）

体制表、チームごとの作業場所数

◆測定方法

現物確認▶体制表、チームごとの作業場所

◆測定データの見方、分かること

・同じ協力会社であっても、インタフェースの密な機能を開発するチームが別々の場所で作業する場合、コミュニケーション不足の可能性は高い。この場合、相互のインタフェースに関する業務分担、責任分担、スケジュールが不明確となり、テストが計画通りに進んでいない可能性がある

モチベーション

★ 62 勤務状況

メンバーの労働時間（残業時間）

◆測定データの属性（参考）

チームごと、個人ごと、月ごとの残業時間推移

◆測定方法

現物確認▶勤務表

◆測定データの見方、分かること

・残業時間数と残業継続月数を見ることにより、メンバー全体の疲労度が分かる。また、どのチーム、どのメンバーの負荷が大きいか分かる（H78）
・負荷の大きいメンバーは離脱の可能性がある（H79）

課題管理

★ 63 作業進捗（課題が増加していないか。）

課題数（未・済）（問題（課題）数（未・済）を変更）

◆測定データの属性（参考）

重要度、発生日、残数
チームごと

◆測定方法

現物確認▶課題管理表

◆測定データの見方、分かること

・課題発生数が増加傾向にある場合、上流工程の設計の品質が悪い可能性がある（H83）
・課題残数が増加傾向にある場合、課題解決パワーが不足している可能性がある。または、解決困難な課題が含まれている可能性がある
・課題解決に要する平均日数が大きい場合、課題解決パワーが不足している可能性がある。または、解決困難な課題が含まれている可能性がある
・課題残数から残作業の負荷、スケジュール遅延の程度が分かる

リスク

★ **64** カットオーバまでのリスクが増加していないか

リスク項目数

◆測定データの属性（参考）

発生数、解消数、リスク発生の確率、発生時の影響の大きさ、リカバリ・プランの有無

◆測定データの見方、分かること

カットオーバーが近づいているにもかかわらず、リスク項目数、発生確率、影響度が増加、あるいは減少していないと要注意である

◆測定方法

現物確認▶リスク管理表

組織

65 協力会社体制の強弱

参画する開発会社数

◆測定データの属性（参考）

◆測定データの見方、分かること

・開発会社数が多い場合、責任の所在が不明確になる可能性がある
・開発会社数が多い場合、会社ごとのセクショナリズムが発生し、各社固有の仕事のやり方で業務が進んだり、コミュニケーション不足になったりする可能性がある

◆測定方法

現物確認▶体制表

組織

66 協力会社体制の強弱

開発会社の階層数

◆測定データの属性（参考）

◆測定データの見方、分かること

開発会社の階層数が多い場合、責任の所在が不明確になる可能性がある（H74）

◆測定方法

現物確認▶体制表

組織

67 協力会社体制の強弱

地理的な開発拠点数

◆測定データの属性（参考）

◆測定データの見方、分かること

拠点数が多くなるとコミュニケーションが悪くなり、相互のインタフェースに関する業務分担、責任分担、スケジュールが不明確になり、テストが計画通りに進んでいない可能性がある

◆測定方法

現物確認▶体制表

組織

68 組織標準への準拠状況

組織標準の順守度合い

◆測定データの属性（参考）

組織標準で定義された作業のプロジェクトの作業への適用率

◆測定方法

現物確認▶標準WBSとプロジェクトのWBS

◆測定データの見方、分かること

プロジェクトのWBSが標準WBSでカバーされている割合が低い場合、プロセスの作業を正確に行えていない可能性がある

組織

69 プロジェクト標準への準拠状況（標準にしたがって作業をしているか）

プロジェクトの標準の順守度合い

◆測定データの属性（参考）

プロジェクト標準で定義された手順を遵守した度合い

◆測定方法

現物確認▶・プロジェクトのWBS
・作業記録

◆測定データの見方、分かること

プロジェクト標準で定義された手順を順守した度合いが低い場合、プロセスの作業を正確に行なえていない可能性がある

組織

★ 70 プロセスの標準への準拠状況（標準にそった管理がされているか）

管理指標数

◆測定データの属性（参考）

品質指標、進捗指標

◆測定方法

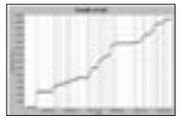
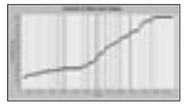
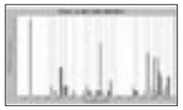
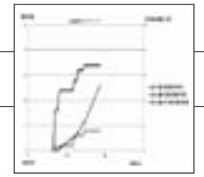
現物確認▶プロジェクトのWBSの単位作業ごとの管理指標

◆測定データの見方、分かること

管理指標がない、あるいはその指標が測定されていない場合、管理されていない可能性がある

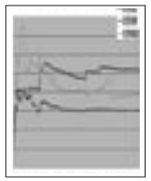
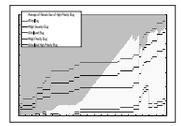
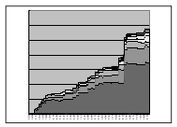
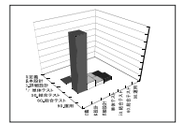
付録 5.EPMツールの分析指標

基本分析指標					
大分類	技術カテゴリ	名称	概要	目的カテゴリ	見える化活用方法
指標	単純プロダクト指標		ソースコード、その他のプロダクトを対象にして測定される属性値	汎用（品質管理、他の複雑な指標のための基礎データ）	
指標例	単純プロダクト指標	ソースコード行数	ソースコード行数	汎用	・より高次の指標のための基礎データとして活用される
指標例	単純プロダクト指標	コメント抜き行数	コメント抜き行数	汎用	
指標例	単純プロダクト指標	サイクロマチック数	コードの複雑さを示す指標	汎用	
指標	EPM 基本推移指標		EPMコアと基本プラグインによって提示可能な指標。時系列に沿った表示形式	汎用	
指標例	EPM 基本推移指標	ソースコード行数推移	時間経過に対するソースコード行数の推移	進捗管理など	・遠隔地であっても開発状況が分かる利点がある ・テスト工程に入ってからの増減から、進捗の状況が分かる（PMOの立場で見ると、プロジェクト・マネージャが増減の理由をきちんと説明できるかで、プロジェクト管理の状況が分かる） ・減っている場合はおかしい（不要コードの削除などを行って整理した、などの理由があればよい）
指標例	EPM 基本推移指標	累積ソースコード変更行数推移	時間経過に対するソースコード変更累積量の推移（単調増加）	進捗管理など	・遠隔地であっても開発状況が分かる利点がある ・テスト工程に入ってからの増減から、進捗の状況が分かる（PMOの立場で見ると、プロジェクト・マネージャが増減の理由をきちんと説明できるかで、プロジェクト管理の状況が分かる）
指標例	EPM 基本推移指標	メール件数の推移	メール件数を時系列にプロットしたもの	プロジェクト管理など	・個人ごと、メール・タイトルごとの送受信件数の推移に異常値があり、その理由を説明できない場合に何らかの問題がある可能性がある
指標例	EPM 基本推移指標	障害発生・解決時期	障害発生時期、解決時期を時系列にプロットしたもの	プロジェクト管理など	・品質の落ち着き具合の判定をできる ・そのプロジェクトの問題解決能力が分かる
指標例	EPM 基本推移指標	チェックアウト頻度	一定期間ごとのチェックアウト回数合計の推移	プロジェクト管理など	・構成管理の仕方（ルール付け）、実践がきちんと行われているかが分かる ・変更される範囲が大きい場合は、品質が悪い可能性がある ・共通モジュールの変更が多いと品質の劣化につながる可能性がある ・バグ票と見比べて、修正作業が正しく行われているかどうかをチェックする
指標例	EPM 基本推移指標	チェックイン契機	一定期間ごとのチェックイン回数合計の推移	プロジェクト管理など	・構成管理の仕方（ルール付け）、実践がきちんと行われているかが分かる ・変更される範囲が大きい場合は、品質が悪い可能性がある ・共通モジュールの変更が多いと品質の劣化につながる可能性がある ・バグ票と見比べて、修正作業が正しく行われているかどうかをチェックする
指標例	EPM 基本推移指標	累積障害件数の推移	累積障害件数の推移	進捗管理、品質管理など	・品質の落ち着き具合を判定できる ・そのプロジェクトの問題解決能力が分かる。テストケースの消化と累積障害件数との関係が重要
指標例	EPM 基本推移指標	累積未解決障害件数の推移	累積未解決障害件数の推移	進捗管理、品質管理など	・品質の落ち着き具合を判定できる ・そのプロジェクトの問題解決能力が分かる
指標例	EPM 基本推移指標	平均障害滞留時間の推移	平均障害滞留時間の推移	進捗管理、品質管理など	・平均障害滞留時間を見れば問題解決能力が分かる ・テスト工数の妥当性が分かる ・総合テストの段階で簡単に修正できるような不良がたくさん出ていれば、単体テストが不十分だったことが分かる

一次データソース	分析ツール	データ収集ツール	自動化レベル	基本測定量	表示例	備考
ソースコード	NA*1	CVS	◎ *2	-	NA	EASE での実装検討中
ソースコード	EPM	CVS	◎	-	NA	
ソースコード	NA*1	CVS	◎ *2	-	NA	EASE での実装検討中
ソースコード	NA*1	CVS	◎ *2	-	NA	EASE での実装検討中
EPM リポジトリ	EPM	CVS,Mailman,GNATS	◎	-	ほぼ任意の組合わせで重畳提示可能	
ソースコード	EPM	CVS	◎	単位時間ごとのソースコード行数		第３章で解説
ソースコード	EPM	CVS	◎	単位時間ごとのソースコード変更量		第３章で解説
メール	EPM	Mailman	◎	単位時間ごとのメール件数		第３章で解説
バグ票	EPM	GNATS	◎	各障害の発生時期、解決時期		
ソースコード	EPM	CVS	◎	単位時間ごとのチェックアウト数		第３章で解説
ソースコード	EPM	CVS	◎	単位時間ごとのチェックイン数		第３章で解説
バグ票	EPM	GNATS	◎	単位時間ごとの障害票件数		第3章で解説
バグ票	EPM	GNATS	◎	単位時間ごとの障害票件数、解決障害件数		第3章で解説
バグ票	EPM	GNATS	◎	単位時間ごとのバグ票件数、解決障害件数		第3章で解説

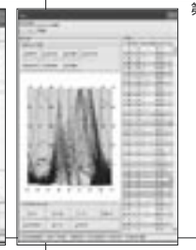
*1 EASE では未提供　*2 EASE では未対応　*3 ライセンス契約などが必要

アドバンスト分析指標					
大分類	技術カテゴリ	名称	概要	目的カテゴリ	
分析手法	協調フィルタリング		過去のサンプル事例を基に未知の値を推定する	汎用	
指標例	協調フィルタリング	ソースコード・ファイルの操作類似度	操作履歴の類似したソースコードを提示	保守管理など	・暗黙の関連を知ることができる（参考：ロジカル・カップリング）
指標例	協調フィルタリング	開発者の類似度	メールのヘッダー情報を基に類似度の高い開発者を提示	プロジェクト管理など	・キーパーソンを推定できる ・組織構成の実態を知ることができる
指標例	協調フィルタリング	メトリクスの類似度	既知のメトリクスに基づいて、あるメトリクスを推定する	プロジェクト最適化など	・工数見積もりやリスク評価値を推定できる
指標例	協調フィルタリング	プロジェクトの類似度	既知のプロジェクト属性に基づいて、あるプロジェクトの属性を推定する	プロジェクト最適化など	・工数見積もりやリスク評価値を推定できる
分析手法	GQM（Goal-Question-Metrics）パラダイム		目的指向の分析ガイド手法。GQM自体は指標ではない。以下はGQMにしたがって導かれる指標例	プロジェクト管理など	
指標例	GQM	プロジェクト遅延リスク検出モデル	構成管理データ（CVS）と障害管理データ（GNATS）のデータからファイル変更パターンを解析し、特定のプロジェクトにおいてプロジェクト・マネージャの視点から不安定な要求、不完全な設計､劣悪なソースコード品質について評価する	プロジェクト管理など	モデルが検出した要求の明確さ、設計の品質やプログラム品質上のリスクを見て、その理由を説明できなければ、問題があることが分かる
指標例	GQM	障害対応進捗状態モデル	バグに関するGNATSデータを分析することによって、特定のプロジェクトにおいてプロジェクト・マネージャの視点から、プロダクトの品質を評価する	プロジェクト管理など	・重要度の高いバグが多いということは、プロジェクトが回っていないことを示している ・プロジェクト遅延要素（リスク）を判断できる。ただし重要度・優先度の基準は設ける必要がある ・特定の個人に偏った状態になっていないかが分かる ・重要度の高いバグ件数より、優先度の高いバグ件数のほうが多い場合は、危険な状態（品質に問題がある）であることが分かる ・優先度の高いバグの件数が減らない場合は、リソース（スキル、人数）に問題がある
分析手法	ロジカル・カップリング		暗黙の依存関係の発見	保守管理など	
指標例	ロジカル・カップリング	ファイル間結合関係	ロジカル・カップリング手法をCVS操作履歴に適用して暗黙のファイル間結合関係を抽出	保守管理など	・データに欠損がある場合でも精度良く予測できる ・予測した品質、工数、工期などに異常値があれば、そこに問題がある可能性がある
分析手法	直交欠陥分類法（ODC）		統計的欠陥モデルと原因分析の間を関連付ける分析方法	プロジェクト分析	
指標例	直交欠陥分類法（ODC）	障害発見工程と平均工数（人時）	バグの混入時期と本来発見できているべき時期を比較する	教育	・発見すべき工程が上流工程にあるケースでは、作業品質が悪いことが分かる ・プロジェクトとしてのロス（手戻り）を定量化できる ・発見すべき工程で発見していない「すり抜け率」が分かる ⇒同一グループに対して適用する（１回目、２回目など）ことによりプロセス改善の程度が分かる

一次データソース	分析ツール	データ収集ツール	自動化レベル	基本測定量	表示例	備考
汎用（ベクトル形式）	NA*3	-	-		データの数量化とベクトル化がの前処理が必要	
ソースコード	NA*3	CVS	△	なし（EPMリポジトリから直接推定）	NA（excelなどによる一覧表示）	第６章で解説
メール and/or バグ票	NA*3	CVS、GNATS など	△	なし（mbox や EPMリポジトリから直接推定）		第６章で解説
各種メトリクスDB	NA*3	各種メトリクス収集ツール	△	各種メトリクス（複数）	NA（excelなどによる一覧表示）	第６章で解説
各種メトリクスDB	NA*3	各種メトリクス収集ツール	△	各種メトリクス（複数）	NA（excelなどによる一覧表示）	第６章で解説
汎用	-	-	-		-	-
構成管理データ、障害管理データ	NA*3	CVS,GNATS	△	CVS 更新回数 CVS 更新量（行数） CVS 更新ファイル数 GNATS 仕様変更数 GNATS 障害数		
障害管理データ	NA*3	GNATS	△	GNATS 障害数 GNATS 障害発生日・対応完了日 GNATS 重要度 GNATS 優先度		
CVS（モジュール操作履歴）	NA	CVS など	-		-	
CVS（モジュール操作履歴）	NA*3	CVS など	△	CVS 同時更新情報（CVS 運用ルールに依存）		
汎用（要因とトリガのペア集合）	NA*3		△			
障害管理データ	NA*3	GNATS	△	GNATS 障害数（GNATS 修正工数・発見日・対応完了日） GNATS 各種分類項目 各種レビュー欠陥データ		SEC で行っている障害分析

*1 EASE では未提供　*2 EASE では未対応　*3 ライセンス契約などが必要

アドバンスト分析指標					
大分類	技術カテゴリ	名称	概要	目的カテゴリ	
分析手法	SPARS		ソフトウェア部品の検索とランク付け	保守管理など	
指標例	SPARS	Java クラス・ファイルのコンポーネント・ランク	ユーザーが入力したキーワードにマッチする部品を検索し、コンポーネント・ランクの順番で出力する	保守管理など	・プロジェクトの中で重要な部品を推定できる
分析手法	コードクローン		コードクローンの検出と特徴分析	保守管理など	
指標例	コードクローン	コードクローンの分布状態	内容が分かっていないソースコード中のコードクローンの分布状態の把握、特徴のあるコードクローンの抽出を行う	保守管理など	計画されたコードクローンの場合 ・スケルトン / 部品化は品質を上げるための要件であり、クローンを使っていないところの量を見ることで、部品化の記述度合いが分かる ・スケルトン / 部品化されるべき部分のクローンが少ないということは、カスタマイズが多いと考えられるため、問題を内在している可能性がある 計画されていないコードクローンの場合 ・コードクローンはないほうがよい。リファクタリングなどによりコードクローンを減し、修正時の影響個所を局在化させる ・個人のコーディングの癖、ウィークポイントの発見も期待される ・問題があったときに、影響範囲が分かる
指標例	コードクローン	リファクタリング可能なコードクローン	Java プログラムからリファクタリング可能なコードクローンを抽出、適用可能なリファクタリング・パターンの提示を行う	保守管理など	・問題の見える化には直接結びつかないが、保守上の課題の見える化に有効

一次データソース	分析ツール	データ収集ツール	自動化レベル	基本測定量	表示例	備考
Java ソースコード	SPARS-J*3	CVS など	-	基本的には部品検索システムである コンポーネント・ランクに基づくソフトウェア内で重要な部品の特定 コンポーネント・ランク計算時に利用している部品間の利用関係を基に保守の影響波及分析にも応用可能		
Java ソースコード（SPARS リポジトリ）	SPARS-J*3	CVS など	○	各部品ごとのコンポーネントランク		コンポーネント・ランク技術は特許申請中（JST 所有）
ソースコード	CCFinder*3	CVS など	-			
ソースコード	CCFinder/ Gemini*3	CVS など	○	-	 	第 6 章で解説
ソースコード	CCFinder/ Aries*3	CVS など	○	-		
ソースコード	CCFinder/ Aries*3	CVS など	○	-		

*1 EASE では未提供　*2 EASE では未対応　*3 ライセンス契約などが必要

付録 6. 症例分類表【一般マップ】

対象 / なにが（なにを）		症状 / どうした	偏り				
			依存	過信	集中・偏在	疲弊	超過
人	個人	プロジェクト・マネージャ（PM）		PMを過信	PMに集中・偏在	PMが疲弊	
		キーパーソン	キーパーソンに依存	キーパーソンを過信	キーパーソンに集中・偏在	キーパーソンが疲弊	キーパーソンが超過
		担当者	担当者に依存	担当者を過信	担当者に集中・偏在	担当者が疲弊	
	グループ	要員			要員が集中・偏在	要員が疲弊	要員が超過
		顧客	顧客に依存				
		協力・関係会社	協力・関係会社に依存				協力・関係会社が超過
	組織	組織					
物	成果物	ドキュメント					ドキュメントが超過
		プログラム		プログラムを過信			プログラムが超過
		仕様		仕様を過信	仕様が集中・偏在		
	材料	ハードウェア	ハードウェアに依存	ハードウェアを過信			
		パッケージ	パッケージに依存	パッケージを過信			
		ツール					
金		予算					予算が超過
		売上					
		費用					費用が超過
スキル	テクニカル・スキル	技術		技術を過信			
		設計					
		テスト			テストが集中・偏在		テストが超過
		見積もり	見積もりに依存				見積もりが超過
		実測					
		知識・経験			知識・経験が集中・偏在		
		レビュー・検証					
		調査					

不十分				変化			
不足	不在	不備（不良）	無関心・無自覚	交代・交換	変更	離脱	低下
PMが不足	PMが不在	PMが不備(不良)	PMが無関心・無自覚			PMが離脱	
キーパーソンが不足	キーパーソンが不在	キーパーソンが不備（不良）	キーパーソンが無関心・無自覚			キーパーソンが離脱	
担当者が不足	担当者が不在	担当者が不備（不良）	担当者が無関心・無自覚	担当者が交代・交換		担当者が離脱	
要員が不足	要員が不在	要員が不備(不良)	要員が無関心・無自覚	要員が交代・交換		要員が離脱	要員が低下
顧客が不足	顧客が不在	顧客が不備(不良)	顧客が無関心・無自覚	顧客が交代・交換			
協力・関係会社が不在	協力・関係会社が不在	協力・関係会社が不備（不良）	協力・関係会社が無関心・無自覚	協力・関係会社が交代・交換		協力・関係会社が離脱	
組織が不足	組織が不在	組織が不備（不良）					
ドキュメントが不足	ドキュメントが不在	ドキュメントが不備（不良）					
プログラムが不足	プログラムが不在	プログラムが不備（不良）			プログラムが変更		
仕様が不足	仕様が不在	仕様が不備(不良)			仕様が変更		
ハードウェアが不足		ハードウェアが不備（不良）					
パッケージが不足	パッケージが不在	パッケージが不備（不良）	パッケージに無関心・無自覚				
ツールが不足	ツールが不在	ツールが不備（不良）					
予算が不足	予算が不在	予算が不備(不良)	予算に無関心・無自覚				
		売上が不備(不良)					
費用が不足	費用が不在	費用が不備(不足)			費用が変更		
技術が不足	技術が不在	技術が不備（不良）	技術に無関心・無自覚				
設計が不足		設計が不備（不良）					
テストが不足	テストが不在	テストが不備（不良）					
見積もりが不足	見積もりが不在	見積もりが不備（不良）					
実測が不足	実測が不在	実測が不備(不良)					
知識・経験が不足	知識・経験が不在	知識・経験が不備（不良）					
レビュー・検証が不足	レビュー・検証が不在	レビュー・検証が不備（不良）					
調査が不足	調査が不在	調査が不備（不良）					

対象 なにが（なにを） ＼ 症状 どうした			偏り				
			依存	過信	集中・偏在	疲弊	超過
	ヒューマン・スキル	基本動作・文化					
		ルール・手順					
		管理					管理が超過
		検討					
		コミュニケーション	コミュニケーションに依存				
		理解・合意					
		注意・考慮					
		判断					
環境		プロジェクト（PJ）内部環境					
		プロジェクト（PJ）外部環境					PJ外部環境が超過
その他		契約	契約に依存	契約を過信	契約に集中･偏在	契約が疲弊	
		計画					
		スケジュール		スケジュールを過信			スケジュールが超過
		範囲・役割					範囲･役割が超過
		ブランド	ブランドに依存	ブランドを過信			
		教育・育成					

不十分				変化			
不足	不在	不備（不良）	無関心・無自覚	交代・交換	変更	離脱	低下
基本動作・文化が不足	基本動作・文化が不在	基本動作・文化が不備（不良）	基本動作・文化に無関心・無自覚				
ルール・手順が不足	ルール・手順が不在	ルール・手順が不備（不良）					
管理が不足	管理が不在	管理が不備（不良）	管理に無関心・無自覚				
検討が不足	検討が不在	検討が不備（不良）					
コミュニケーションが不足	コミュニケーションが不在	コミュニケーションが不備（不良）	コミュニケーションに無関心・無自覚				
理解・合意が不足	理解・合意が不在	理解･合意が不備（不良）					
注意・考慮が不足	注意・考慮が不在	注意・考慮が不備（不良）					
判断が不足	判断が不在	判断が不備（不良）					
PJ内部環境が不足	PJ内部環境が不在	PJ内部環境が不備（不良）					
PJ外部環境が不足	PJ外部環境が不在	PJ外部環境が不備（不良）					
契約が不足	契約が不在	契約が不備（不良）			契約が変更		
計画が不足	計画が不在	計画が不備(不良)					
スケジュールが不足	スケジュールが不在	スケジュールが不備（不良）			スケジュールが変更		
範囲・役割が不足	範囲・役割が不在	範囲･役割が不備（不良）			範囲・役割が変更		
		ブランドが不備（不良）					
教育･育成が不足	教育･育成が不在	教育･育成が不備（不良）					

付録 6. 症例分類表【ヒアリングマップ】

対象 なにが（なにを）		症状 どうした	偏り 依存	偏り 過信	偏り 集中・偏在	偏り 疲弊	偏り 超過	不十分 不足
人	個人	プロジェクト・マネージャ		H72				H62,H72
		キーパーソン						H3,H43,H47,H63
		担当者	H7	H7	H7			
	グループ	要員			H17	H78	H19,H21	H19,H21
		顧客						H43,H62
		協力・関係会社						H58
	組織	組織						H41,H45
物	成果物	ドキュメント						H2,H6,H13,H31,H38, H46,H50,H51,H53,H56, H61,H64,H65,H76, H83,H84
		プログラム						H6,H38,H70,H71
		仕様						H2,H70,H71
	材料	ハードウェア						H70,H71
		パッケージ	H67	H67				H67
		ツール						H4,H6,H27,H29, H70,H71
金		予算						H3
		売上						
		費用					H24	H12,H24
スキル	テクニカル・スキル	技術		H68				H58,H68
		設計						H2,H70,H71
		テスト						H2,H14,H26,H27,H29, H30,H32,H33,H35, H36,H37,H49,H70, H71
		見積もり						H3,H16,H21,H25, H54,H58
		実測						H22,H34,H44,H54
		知識・経験						H2,H3,H29,H58
		レビュー・検証						H9,H28,H29,H33,H34, H39,H43,H46,H47,H49, H50,H51,H67,H69,H84
		調査						H21,H25,H65, H67,H68

不十分 不在	不十分 不備（不良）	不十分 無関心・無自覚	変化 交代・交換	変化 変更	変化 離脱	変化 低下
	H62,H81	H17,H79,H80				
H43,H47,H63	H43,H47,H63	H80				
H7,H83	H7	H7,H80,H83	H7,H13		H7	
	H1	H80	H79		H79	H78
H43,H62	H43,H62,H66	H62,H80				
H58	H74,H58	H80				
H41,H45	H41,H42,H45, H72,H74					
H2,H13,H31,H38, H46,H50,H51,H53, H56,H61,H65,H76, H83,H84	H6,H13,H31,H38,H46, H50,H51,H53,H56, H61,H64,H65, H76,H83,H84					
	H6,H28,H38, H39,H70,H71			H39		
	H2,H70,H71					
	H70,H71					
	H67					
H6,H29,H70,H71	H4,H70,H71					
	H3	H3				
	H15					
				H12,H24		
H58,H68	H58,H68	H68				
	H70,H71					
H14,H49,H70,H71	H4,H14,H26,H27, H29,H30,H32,H33, H35,H36,H37,H49, H70,H71					
H54,H58	H3,H5,H16,H21, H25,H54,H58					
H34,H44	H22,H34,H44, H54					
H3,H29,H58	H3,H58					
H9,H28,H33,H34, H39,H43,H46,H47, H49,H50,H51,H67, H69,H84	H9,H28,H33,H34, H39,H43,H46,H47, H49,H50,H51,H67, H69,H84					
H65,H67,H68	H65,H67,H68					

対象 なにが（なにを）		症状 どうした	偏り					不十分
			依存	過信	集中・偏在	疲弊	超過	不足
スキル	ヒューマン・スキル	基本動作・文化						H1,H11,H22,H77
		ルール・手順						H6,H8,H9,H11,H15, H28,H30,H32,H37, H38,H39,H42,H44, H48,H52,H60,H75, H76,H85
		管理						H2,H6,H8,H9,H11, H28,H31,H34,H38, H48,H57,H75,H78, H79,H83
		検討						H2,H5,H23,H25,H47, H48,H55,H58,H59, H66,H69,H70
		コミュニケーション						H1,H5,H9,H13,H20, H31,H32,H40,H41,H42, H44,H45,H48,H52, H57,H74,H76,H81,H82
		理解・合意						H5,H12,H13,H14,H15, H20,H43,H44,H46, H47,H49,H50,H51, H57,H61,H66,H81, H84
		注意・考慮						H55,H63,H77, H78,H79
		判断						H59,H61,H85
環境		プロジェクト内部環境						H4,H35,H78
		プロジェクト外部環境						H78
その他		契約						H5,H15,H53,H60, H61,H64
		計画						H9,H10,H32,H33,H35, H36,H37,H44,H57,H58, H66,H69,H82
		スケジュール					H19	H2,H3,H5,H16,H17, H50,H51,H64
		範囲・役割						H5,H14,H16,H17,H23, H44,H51,H62,H64, H65,H66,H69,H80
		ブランド	H67	H67				
		教育・育成						H82

不十分			変化			
不在	不備（不良）	無関心・無自覚	交代・交換	変更	離脱	低下
H1,H77	H1,H11,H22, H42,H77	H56				
H1,H6,H8,H9H11, H28,H32,H38,H39, H48,H52,H60,H75, H76,H85	H5,H6,H8,H9,H11, H28,H30,H32,H37, H38,H39,H44,H48, H52,H60,H75,H76, H85					
H2,H6,H8,H9,H11, H28,H31,H34,H38, H48,H57,H75,H78, H79,H83	H2,H6,H8,H9,H11, H22,H28,H31,H34, H38,H48,H57,H75, H78,H79,H83					
H48,H58,H59, H66,H69,H70	H5,H47,H48,H58, H59,H66,H69,H70					
H9,H45,H48,H52, H57,H76,H81,H82	H1,H9,H13,H42, H44,H45,H48,H52, H57,H76,H81,H82					
H5,H43,H46,H49, H50,H51,H57, H61,H66,H81,H84	H5,H12,H13,H14,H15, H43,H44,H46,H47,H49, H50,H51,H57,H61,H66, H81,H84					
H63,H77,H78,H79	H63,H78,H79					
H59	H55,H59,H61,H85					
H4,H35	H4,H35,H78					
	H78					
H53,H60,H61	H3,H5,H15,H24,H53, H60,H61,H64,H74			H24		
H32,H57,H58, H68,H69,H82	H9,H10,H26,H30,H32, H33,H35,H36,H37,H44, H57,H58,H66,H68, H69,H82					
H50,H51	H1,H5,H14,H16,H17, H18,H19,H20,H21,H23, H26,H30,H50,H51,H64			H12		
H69,H80	H5H14,H16,H17,H23, H40,H41,H44,H51,H62, H64,H65,H66,H69, H73,H74,H80			H11,H12		
H82	H82					

付録 6. 症例分類表【測定項目マップ】

対象 / なにが（なにを）		症状 / どうした	偏り					不十分				変化			
			依存	過信	集中・偏在	疲弊	超過	不足	不在	不備（不良）	無関心・無自覚	交代・交換	変更	離脱	低下
人	個人	プロジェクト・マネージャ						49		49					
		キーパーソン						47	47	47					
		担当者			54,55,56,57			47,54,55,56,57	47,54,55,56,57	47,54,55,56,57					
	グループ	要員			13,28,52,62	13,28,62	22,62	13,22,28,48,51,52,62	51,62	13,22,26,28,48,51,62		62		62	62
		顧客													
		協力・関係会社					65,66	65,66	65,66	26					
	組織	組織						54,55,56,57	54,55,56,57	54,55,56,57					
物	成果物	ドキュメント					3	3,17,18	3,17,18,	3,17,18,29					
		プログラム					4,40,41	4,40,41	4,40,41	4,15,16,25,26,28,40,41					
		仕様						1,2,10		1,2,25,28			1,2		
	材料	ハードウェア													
		パッケージ								25,28					
		ツール													
金		予算					22,24	22,24	22,24	22,24					
		売上													
		費用					21,22	21,22	21,22	21,22					
スキル	テクニカル・スキル	技術						12,28,42,48,49,58	12,28,42,48,49,58,	12,28,42,48,49,58,					
		設計						10,50		50					
		テスト			43		43	20,25,32,43,44,51	20,25,32,43,44,51	20,25,32,43,44,51					
		見積もり					4,21,24,40,41	4,21,24,40,41	4,21,24,40,41	4,21,24,40,41					
		実測						6,9,11,12,13,42,70	6,9,11,12,13,42,70	6,9,11,12,13,42,70					
		知識・経験						12,28,42,48,49,50,51,58	12,28,42,48,49,50,51,58	28,48,49,50,51,58					
		レビュー・検証						34	34	34					
		調査						30,31	30,31	30,31					
	ヒューマン・スキル	基本動作・文化						59	59	59					
		ルール・手順						1,2,14,29,30,31,34,68,69,	1,2,14,29,30,31,34,68,69	1,2,14,29,30,31,34,68,69					
		管理					65,66	14,25,29,59,62,63,64,65,66,67,69,70	59,62,63,64,69,70	14,15,16,17,18,25,29,59,62,63,64,65,66,67,69,70					
		検討													
		コミュニケーション						59,67	59	59,67					
		理解・合意													
		注意・考慮						44	44	44					
		判断													
環境		プロジェクト内部環境						19,20,61,67	19,20,61,67	19,20,61,67					
		プロジェクト外部環境													
その他		契約													
		計画						10		10					
		スケジュール					22	6,9,11,22	22	6,9,11,22					
		範囲・役割					15,16,17,18,52,63,65	6,15,16,17,18,32,44,52,63,65,66,68	32,44,68	6,15,16,32,44,52,63,65,66,68			63		
		ブランド													
		教育・育成													

付録 6. 症例分類表【事例マップ】

対象 なにが（なにを）		症状 どうした	偏り 依存	過信	集中・偏在	疲弊	超過	不十分 不足
人	個人	プロジェクト・マネージャ		17,45	15,19,35	59		12,63,72
		キーパーソン	41,64	4	64	9,64	9	
		担当者	1		66	45,48, 66,74		40,58
	グループ	要員			37,43			18,37,43
		顧客	24					20,48,60
		協力・関係会社	28					1,52,65
	組織	組織						
物	成果物	ドキュメント						3
		プログラム		33,56				3
		仕様		29	6,17,63			13,14,49,73
	材料	ハードウェア	77	42				
		パッケージ	54,57, 61,75	23,29,57, 61,75				5,55,61
		ツール						
金		予算						
		売上						
		費用						
スキル	テクニカル・スキル	技術						6,10,26,30,38,43, 54,58,67,72,76
		設計						25,27,39
		テスト						25,29,32
		見積もり	12					11,12,22,29
		実測						
		知識・経験			39			17,18,29,43
		レビュー・検証						76
		調査						11,10,13,14,17
	ヒューマン・スキル	基本動作・文化						12,44,51,76
		ルール・手順						
		管理						58
		検討						20
		コミュニケーション	46					7,8,10,12,17,44, 45,47,48
		理解・合意						24,25,44
		注意・考慮						44
		判断						
環境		プロジェクト内部環境						
		プロジェクト外部環境					13	
その他		契約	10	16,42	16	20		16
		計画						
		スケジュール		70				29
		範囲・役割						
		ブランド	57					
		教育・育成						18,38,50,64

不十分 不在	不備（不良）	無関心・無自覚	変化 交代・交換	変更	離脱	低下
12,41,58	24,34,67	57			52	
	9				9,41	
34,40,42,69	34,45,50,62	39,45,58,69			1,52	
14,50,60,71	60	60	50			
		28,	51,52		51	
3,31,46	31					
	21,76,77,78					
14,49	2,6,17,31,73					
	77					
56	29,57	29,56				
30,36						
30	25					
24	11					
78						
17	10,14,24,25,31	44,55,69				
12,13,20,21,22, 25,58	24	28				
4,6,9,40,44	15,17,40,45,47, 48,58,62	44				
44						
44						
	68					
13						
53	16,17,19,20,42					
11,19,24,70	1,25,63,67,68					
	29,57					

参考文献

・遠藤 功：”現場力を鍛える「強い現場」をつくる7つの条件”,東洋経済新報社,2004
・遠藤 功：”見える化-強い企業をつくる「見える」仕組み”,東洋経済新報社,2005
・プロジェクトマネジメント協会：”プロジェクトマネジメント知識体系ガイド(PMBOKガイド)第3版”,2004
・Capers Jones：”ソフトウェア病理学―システム開発・保守の手引”,(株)構造計画研究所,1995
・カール・E・ウィーガーズ(著),滝沢 徹(翻訳),牧野 祐子(翻訳)：”ソフトウェア開発の持つべき文化IT Architects' Archiveソフトウェア開発の課題”,(株)翔泳社,2005年6月
・松本 健一: “エンピリカル・ソフトウェア工学の現状と展望：SELが遺した13の教訓”,SEC Journal,2005/4/25
・大杉直樹,松村知子,森崎修司：“ソフトウェア開発の「見える化」を支援するデータ分析力 ～エンピリカルアプローチによる既存データの有効活用～”,JISA会報,No.80,2006年1月
・井上 克郎,松本 健一,鶴保 征城,鳥居 宏次: “実証的ソフトウェア工学環境への取り組み”,情報処理7月号,vol.45,No.7,pp722-728,2004.,2004/7/15
・樋口 登：”先進ソフトウェア開発プロジェクト”：SEC journal2号,2005年4月,pp.56-57
・松浦 清、神谷 芳樹、樋口 登：”先進ソフトウェア開発プロジェクト Part II”,SEC journal 5号, 2006年1月, pp.44-49
・情報処理推進機構(編)：”ソフトウェア開発プロセスの「見える化」、車社会を変えるプローブ情報システムの開発(COSE)”,柔の力、剛の技 第3部, アスキー, 2006年5月, pp.160-176
・ 神谷芳樹：”EASEプロジェクト”,SEC journal 1号, 2005年1月, pp.40-41
・ 神谷芳樹：”EASEプロジェクトに見る計測・定量化の実践”, 日経ITプロフェッショナル, 2005年3月, pp.92-97
・井上克郎、松本健一、鶴保征城、鳥居宏次：”実証的ソフトウェア工学環境への取り組み”, 情報処理, vol.45, No.7, 情報処理学会, 2004年7月, pp.722-728
・John McGarry, David Card, Cheryl Jones,Beth Layman, Elizabeth Clark, Joseph Dean, Fred Hall(著),古山恒夫・富野壽(訳)：”実践的ソフトウェア測定”, 共立出版, 2004
・Stephen H. Kan(著), 古山恒夫・富野壽(訳)：”ソフトウェア品質工学の尺度とモデル”, 共立出版, 2004
・野村総合研究所 システムコンサルティング事業本部：”最新 図解CIOハンドブック”, 野村総合研究所広報部, 2005
・独立行政法人情報処理推進機構, ソフトウェア・エンジニアリング・センター：”ソフトウェア開発データ白書〈2005〉”, 日経BP社, 2005年5月
・ 独立行政法人科学技術振興機構(JST)(統括・畑村 洋太郎)：”失敗知識データベースの構造と表現(「失敗まんだら」解説)”,
http://shippai.jst.go.jp/fkd/Contents?fn=1&id=GE0704, 2005年3月
・ ソフトウェアCALS実証プロジェクト：”ソフトウェアCALS環境構築用のソフトウェア開発及び実証”,ソフトウェアCALS実証コンソーシアム, 1998年7月,
http://www.ipa.go.jp/archive/NBP/CREC/EC_List2.htm

執筆者

プロジェクト見える化部会委員（◎：主査、○：副主査）

○	秋山　雅俊	株式会社協和エクシオ
	飯田　元	国立大学法人奈良先端科学技術大学院大学
	木曽　晋也	ソフトウェア・エンジニアリング・センター（三菱電機株式会社）
	木根　秀隆	日立ソフトウェアエンジニアリング株式会社
	栗田　存	株式会社クロスリンク・コンサルティング
	香村　求	株式会社システムSWAT
◎	長岡　良蔵	ソフトウェア・エンジニアリング・センター（日立ソフトウェアエンジニアリング株式会社）
	中村　修	株式会社 IHI エスキューブ
○	西川　広	新日鉄ソリューションズ株式会社
	拝原　正人	株式会社クロスリンク・コンサルティング
	樋口　登	ソフトウェア・エンジニアリング・センター（日本電気株式会社）
	神谷　芳樹	ソフトウェア・エンジニアリング・センター
	安田　守	ソフトウェア・エンジニアリング・センター（株式会社野村総合研究所）

研究解説執筆、編集等協力

大杉　直樹	EASEプロジェクト（国立大学法人 奈良先端科学技術大学院大学）
阪井　誠	EASEプロジェクト（株式会社SRA先端技術研究所）
楠本　真二	EASEプロジェクト（国立大学法人 大阪大学）
肥後　芳樹	EASEプロジェクト（国立大学法人 大阪大学）
遠藤　潤	株式会社野村総合研究所
小久保　岩生	株式会社三菱総合研究所
新田　一樹	株式会社野村総合研究所

*第6章の6.1、6.3、6.5および付録5の内容の一部はEASEプロジェクト（文部科学省リーディングプロジェクト e-Society基盤ソフトウェアの総合開発）の成果に基づくものです

ITプロジェクトの「見える化」 下流工程編

2006年6月19日	初版第1刷発行
著作／監修	独立行政法人 情報処理推進機構（IPA） ソフトウェア・エンジニアリング・センター（SEC）
編集	日経コンピュータ
発行人	藤田 俊一
発行	日経BP社
発売	日経BP出版センター 〒108-8646　東京都港区白金1-17-3 TEL（03）6811-8200
表紙デザイン	成田 美由喜（日経BPクリエーティブ）
制作	日経BPクリエーティブ
印刷・製本	大日本印刷

ISBN4-8222-6201-4

日経BP社
Nikkei Business Publications, Inc.
〒108-8646　東京都港区白金1-17-3