失敗しないシステム開発マニュアル

第1回

ユーザーがトラブルに巻き込まれない心構え7つのポイント

牧野総合法律事務所弁護士法人（JIIMA顧問弁護士）　牧野（まきの） 二郎（じろう）

今にち、企業の業務遂行において、コンピュータシステムを利用することは不可欠となっています。その一方で、自社の業務にぴったりと合ったシステムはなかなか見つからないものです。そこで、ベンダーにシステム開発を依頼することになるのですが、ユーザーはコンピュータシステムのことについては素人で、その内容を理解しておらず、他方、ベンダーはユーザーの業務のことについて理解しておらず、両者の理解・認識に差があるということがよくあります。

システムに疎いユーザーと業務に疎いベンダー、両者が分かりあえないことによって、かなりの高確率でシステム開発はトラブルに巻き込まれてしまいます。今回は、こうしたトラブルに巻き込まれないために、主にシステム開発のプロジェクト責任者に向けて、注意すべきポイントや心構えについて解説します。

システム開発が必要な企業の背景

現代において、いかなる企業にとってもシステム開発は避けて通れない問題となりつつあります。

企業は事業活動を通して、利益を上げ、成長拡大する必要があります。そのためには、無駄を省き、正確に業務処理できる体制を固め、効率的経営を進めることが必須となります。コンピュータシステムには、こうした業務処理の一端を担い、休むことなく処理を続ける力があります。大変便利な道具です。こうした便利な道具を使わずに企業が成長することは考えにくいのです。したがって、システム開発はほとんどの企業に共通する課題であり、システム開発に向き合うすべての人の問題になるのです。

システム開発で目指すべき業務の効率化とは？

業務の効率化は業務処理時間を短縮し、作業効率が向上することで収益増加の可能性が飛躍的に上がります。効率的な業務を行うことは、同種の業務を行う他の企業よりも原価率を下げることになり、利益の向上にもつながり、競争力をもつことになります。

そこで、「業務効率化のために新たにシステムを開発・導入すれば良い」と単純に考えがちですが、短絡的に考えるのはトラブルにもつながりやすくなり、システム開発が頓挫すれば、かえってあらぬ出費を招いてしまいます。また、あまりに複雑で非効率な業務工程を、そのままシステムに落としこんでみたところで、そのシステムは、複雑で非効率な業務工程を固定化してしまうだけで、業務が効率化することにつながりません。システムにバグが発生する危険も高くなり、システム化の意味を果たさないのです。

コンピュータシステムを導入したら成功するはず、システム化したら必ず今の業務は効率化するだろうという幻想は、この際、きっぱり捨ててください。まずは、業務そのものを見直すことが

<システム開発のプロジェクト責任者の心構え7つのポイント>

①	何を作りたいのかを明確にし、文章化する。	――明確な目標設定
②	現在の業務を精査し、業務改善を行う。	――業務手順見直し
③	システムに任せる部分と、人の作業を明確に区分する。	――業務分析
④	開発事業者を雰囲気で決めるな。	――業者選定
⑤	言語の違い、経験の違いを理解せよ。	――概念整理
⑥	理解するまで前に進むな。判子を押すな。	――検収の重要性
⑦	随時協議し、記録を残せ。	――記録管理

大事なのです。例えるならば、汚い部屋を写真にとっても汚いだけです。綺麗な部屋にしてから写真を撮るべきなのです。

「働き方改革」を進めるためのシステム

わが国では、「働き方改革」が大いに話題となっていますが、無理な改革は生産性の低下につながり、ややもすれば外形だけの改革に終わりかねません。単純に労働時間を短縮したり、自宅勤務を認めたら、作業効率は下がります。それをカバーしようとする管理職の過労が進むことも考えられます。本来の「働き方改革」は、業務の合理化を図り、人の担う部分を可能な限り低減し、作業効率を犠牲にせず、むしろ働きやすい環境の下でより集中して、より働き甲斐のある仕事ができるようにする、ということのはずです。

そのために何が必要なのでしょうか。自宅勤務を進めるためにはセキュリティ対策が必須ですし、会社の同僚とのコミュニケーションの確保も必須です。そのためのインフラの整備、システムの構築はなくてはならないものとなります。こうした環境の整備、インフラシステムの導入をする際にはその会社の業務や現状に見合うものとする必要があります。

デジタル社会におけるシステム導入のあり方

いまや、一部の企業では、IoT（モノのインターネット化）を活用し、そこから得るビッグデータを分析し活用することが行われています。わが国は、センサーの分野において世界の最先端を走っているといわれていますが、そのセンサーが生み出すデータの利活用には、効率的なシステム、賢いAI（人工知能）が必要となります。金融、金流の世界にもフィンテック（FinTech）が広がり、システム化、AIの導入が進んでいます。

しかし、こうした社会の変化に翻弄されて、やみくもにシステムの導入を進めても、膨大な費用を回収できるほどに利益を生み出せるかは疑問です。AIもまだまだ活用できるレベルではないといわれていますし、そもそも、AIを使いこなせる技術者が足りないので、その導入には十分な準備と、体制整備が求められるのです。

システム開発の失敗には必ず理由がある

こうした現在の社会的変化、企業を取り巻く環境の変化に対応する必要があることから、いずれの企業も、我先にとシステムを導入して効率化を進めようと考えるわけですが、そこにはいくつもの大きな落とし穴があります。

システム開発は、成功する確率より失敗する可能性の方が高いとよく言われます。システム開発の失敗には必ず原因があります。そしてその原因は、失敗したシステム開発を見てみると、かなりの共通点があるのです。

まず、私の経験から申し上げると、失敗する典型例は次のようなものです。ベンダー企業のプレゼンにユーザー企業の担当者が惹きつけられ、綺麗なコンピュータの画面が描かれたパワーポイントを見て、ユーザー企業の皆さんは「素晴らしいシステムができるに違いない」と信じてしまいます。そして、契約書のほか重要な書類に次から次へと調印し、実際にシステムの稼働が近づいてきた時点で、自社の業務に全く使えないシステムが開発されていたことに気付く、というものです。

このような失敗はあくまで顕著な例ですが、似たような失敗例は私も多く見てきました。こうした失敗は多くの場合、ユーザー企業側の心構えで防ぐことができます。以下では、これから開発作業に取り掛かる企業の担当者が持つべき心構えの7つのポイントを解説します。

①何を作りたいのかを明確にし、文章化する。
――――明確な目標設定

システム開発を考える際には、現在の業務を改善したい、新しいサービスを展開したい、などさまざまな動機があるはずです。ただ、これは動機であって、主観的なものに過ぎません。システム開発は、全社の各階層のメンバーが理解し、かつ開発作業を実施する外部の事業者も正確に認識できるものでなければなりません。

そこで、必要になるのが「明確な目標の設定」です。「改善したい」というその具体的な内容は何か。スピードアップなのか、省力化なのか、それとも品質向上なのか、安全性確保なのか、ユーザー側で明確にしておかないと、ベンダー側の受け取り方で変わってしまうでしょう。正確な目標設定ができなければ、まともな開発はできませんし、出来上がったものの評価もできず、結局成功不成功の判断すらできないまま、開発は失敗に終わるのです。まずは、明確な目標設定、しかも正確な文字にして、伝えることが重要です。これをRFP（提案依頼書）と言います。

②現在の業務を精査し、業務改善を行う。
――――業務手順見直し

新規にシステムを開発するといっても、現業業務と無関係の

ものを作るわけではなく、現業業務を基礎にしたもののはずです。現業業務のまま、システム化した場合には多少の省力化(省人化)が図れるかもしれませんが、かえって効率性が下がったり、システムトラブルに巻き込まれたりするはずです。これも開発の失敗です。

開発の前に、現業業務を見直さなければなりません。洗い出しをして、合理的で、無駄のない業務の流れにして、それをシステム化することで、無駄のないシステムができて、効率が上がることになります。実際に、こうした業務手順改善をしただけで、業務の効率は向上するのです。

③システムに任せる部分と、人の作業を明確に区分する。――業務分析

システムは、人なしでは動きません。入力作業、点検作業、出力データに従った業務処理など、多数の人力との連携が重要です。いくらスピードアップできるシステムでも、入力が複雑になり、アウトプットが業務工数を増加させたのでは本末転倒です。

徹底した現業業務分析、想定されるシステム完成後の業務分析を実施して、本当に業務改善がなされているのか、人とシステムの連携がスムーズなのかを検討しなければなりません。

④開発事業者を雰囲気で決めるな。――業者選定

いよいよ開発システムの発注ですが、多くのユーザー企業は、値段や工期を優先させているようです。最近ではWEBでの評価を重視しているという話も聞きますが、多くの場合は「見た目8割」で業者を選定してしまっています。これでは開発が失敗するのは当然なのです。

開発業者を選定する場合、最後まで責任をとる開発者なのか、技術者をアサインできるのか、その技術者の経験値はどうか、発注者の意見や希望を丹念に聞いてもらえるか、疑問や課題をはっきり説明できるか、など多くの観点から判断する必要があります。良い仕事をしてくれる開発者は引っ張りだこです。ここが成功と失敗の分かれ目といってもいいでしょう。

⑤言語の違い、経験の違いを理解せよ。――概念整理

どんなに優秀なエンジニアがついてくれたとしても、安心はできません。なぜなら、彼はあなたの業務の本質をまだ知らないからです。システム開発のプロは、ユーザー業務については素人という場合が往々にしてあります。

エンジニアは、業務の基本的な内容を知りませんから、そこから説明し、理解してもらう必要があります。ここで遠慮したり、知ったかぶりをすると、後で大失敗を起こすことになります。慎重に、相互の理解度を確認しながら、進めなければなりません。

⑥理解するまで前に進むな。判子を押すな。――検収の重要性

ユーザーの失敗ポイントは、契約書、検収書に、安易に押印してしまうことです。期限を設定され、意味も分からず押印してしまうのです。

契約は、これから作るものの内容を明確に示しているのか、その姿をどのように示すのか、問題があったら改善できるのかなど、いくつもの課題をクリアしなければなりません。

検収は、要件定義書、仕様書、納品書、システム納入などの際に求められますが、これは「問題ありません。あとは私の責任とします」という約束なのです。納入されたものが本当に問題がないのか、予定されたものか、要求が網羅されているか、欠陥はないかなど徹底的な検証をしない限り、押印してはいけないのです。これを安易にすると、後戻りができません。

⑦随時協議し、記録を残せ。――記録管理

ここでの最大のポイントは、「すべて記録せよ」という点になります。文書記録が、システム開発においても、絶大な威力を発揮します。打合せの段階でいろいろな要求、必要な機能の話が出ますが、これを記録して議事録にしておかないと、後になって、言った、言わない、約束した、しない、の議論になり、トラブルの元です。

徹底した記録は、開発業務のすべての当事者に良い影響を及ぼします。合意内容が記録されているので、全員がその記録を確認し、記録に従って作業をして、さらにその作業結果の記録を行い、訂正や修正も記録して進める、ということになります。こうして正確な記録は、全体を取り仕切る管理者のような役割を果たしてくれるのです。

記録管理をしないシステム開発は、管理者のいない開発になり、失敗する運命を負わされます。

まとめ

以上の7つのポイントを心構えとしていただければ、システム開発のトラブルの多くは回避できると思います。

第2回以降は、システム開発トラブルの実例やプロジェクト管理の実践的な作業を解説していくことで、トラブルを回避するマニュアルを皆さんと共有していきます。

第2回

スルガ銀行・日本IBM巨額泥沼訴訟の失敗例から学ぶベンダーの心得

牧野総合法律事務所弁護士法人（JIIMA顧問弁護士）　牧野（まきの） 二郎（じろう）

前回（第1回）は、主にユーザー側のシステム開発プロジェクト責任者に向けて、注意すべきポイントや心構えについて解説しました。

システム開発は、ベンダーとユーザーが協力して進めるものです。前回は主にユーザー側から見たので、今回はベンダーに注目してみます。ベンダーの義務、やるべきことはベンダーの心得でもありますが、ユーザーとしてもベンダーが行うべき作業をきちんと実施しているかをしっかり見据えて、進行をチェックする必要があります。

今回は、システム開発訴訟で最も重要な事件の一つである「スルガ銀行・日本IBM事件」を見ながら、システム開発の失敗の原因を探ります。

「スルガ銀行・日本IBM事件」は、システム開発をめぐって泥沼の巨額訴訟にまで発展したシステム開発の「失敗例」です。日本IBMは、ベンダーとしても巨大な組織を持つ専門家集団のはずです。その専門家集団ですら、基礎を踏み外し、とんでもない失敗を招いてしまったという実例を取り上げてみます。平成時代の最大ともいうべき「失敗例」からベンダーの心得を解説します。

スルガ銀行・日本IBM事件のあらすじ

スルガ銀行・日本IBM事件は米国製の勘定系パッケージソフトをカスタマイズする開発が頓挫したというものです。まずは、この事件のあらすじを見てみましょう。

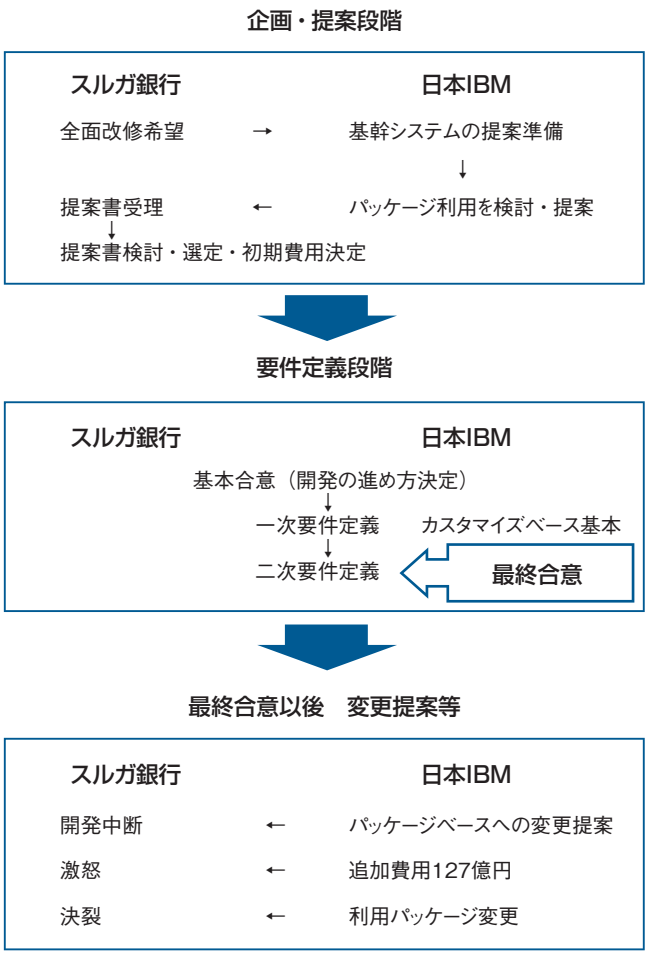
あらすじとしては、「企画・提案段階」「要件定義段階」「最終合意以降」の3つの段階に分けて把握することが必要になってきます。

この開発案件は、紆余曲折があり、事実関係は大変複雑です。しかし、その個々の事実に引きずられることなく、システム開発の基本の流れ（開発段階）を見据えておく必要があります。すなわち、①企画・提案段階、②要件定義段階、③最終合意段階という3つの大きな段階に分解します。この段階区分は、開発当事者の合意に基づいており、裁判はこの「合意の成立」に着眼したものといえます。これほど巨大な開発も結局、当事者の合意が基礎となることを理解してください。

合意形成の概要をまず図式してみます。

スルガ銀行がシステムの全面改修を希望し、それに応える形で日本IBMが対応し開発作業がスタートします。

大まかに見てみると、要件定義までは順調のようですが、最



「スルガ銀行・日本IBM事件」の流れ

後の最終合意以降の開発側の日本IBMに、目に余るぐらつきがあるように見えます。

しかし、この日本IBMのぐらつきは、実は当初の提案から安易な見込み、企画段階から始まっていたことが分かります。

以下、検証します。

企画・提案段階

日本IBMは30年以上にわたってスルガ銀行のシステムの管理、運用の支援、OSの保守等を行ってきました。30年の理解の蓄積があるとみるか、30年経過して新しい時代になり、それまでの経験は役立たなくなっているとみるかが一つのポイントです。

スルガ銀行としては、昭和46年10月に導入した基幹システムの老朽化が進んできたことなどの理由から、その刷新を図るため、平成12年ごろ日本IBMに対し、基幹系システム構築の提案及び同システムの運用、開発について検討を依頼しました。

日本IBMは、平成14年1月に次世代の勘定系ソリューションの一つとして「NEFSS[1]」を企画、開発するためのチームを立ち上げ、米国企業が権利を保有していたパッケージソフトCorebankをNEFSSの部品として採用する方向で検討を開始しました。

そして日本IBMは、平成16年3月12日、スルガ銀行に対し、「短期間でパッケージを構築するには、ゼロからの開発ではなく実績のあるパッケージを有効活用して開発する方が優位である」などと記載した提案書を交付しました。

スルガ銀行は、日本IBMのほか、複数のシステム開発業者から提案されたシステムを比較検討した結果、日本IBMによる提案の内容が最も好ましいものと考え、平成16年9月21日、初期費用約95億円を支出することを決めました。

この時点で、スルガ銀行の判断が適切であったかにも疑問があります。スルガ銀行にはシステム部門があったのですから、既存の銀行業務と提案されたパッケージとの違いを明確にして対応を検討すべきであったのは明らかです。このスルガ銀行の安易な判断が、日本IBMの思い込みと独走を許してしまった原因でもあるでしょう。

要件定義段階

① 基本合意

スルガ銀行と日本IBMは、平成16年9月29日、95億円でシステム稼働を実現することなどが記載された「新経営システム」構築に関する基本合意書①を締結しました。

このプロジェクトにおいては、原則として毎月1回、スルガ銀行及び日本IBMの現場の開発担当者から役員までを出席者として会議「ステアリング・コミッティ」を行うこととされ、このステアリング・コミッティは、平成16年11月5日に第1回目が開催されて以来、平成19年3月まで合計27回開催されることになります。

② 一次要件定義

平成16年9月30日～12月28日の要件定義においては、本件プロジェクトの目標、開発対象となるシステム及び開発スケジュールの明確化、開発方法の検討などが行われ、日本IBMはスルガ銀行に対し、成果物として、12月21日付けプロジェクト管理計画書「PMP」を作成しました。「要件定義」とは、システムの設計前にシステム化のための要件を洗い出して確定する作業のことです。この段階では、その前段が定められたに留まります。

③ 二次要件定義

平成16年12月29日には、2回目の要件定義を行うこと、要件定義書を成果物として納品すること、それまでの確定金額は35億円であることなどを記載した「新経営システム」構築に関する基本合意書②を締結しました。

スルガ銀行と日本IBMは、平成16年12月29日、2回目の要件定義を開始しましたが、この要件定義においては、本件システムにおいて実現すべき「変革のテーマ」等をどのように実際に具現化していくかについての方法の検討、本件システムに継承すべき現行システムが有する機能の分析、スルガ銀行の現行システムとNEFSS/Corebankを用いたシステム上の処理との差異の分析（いわゆるFIT&GAP分析）が行われました。

なお、スルガ銀行の現行システムが有する機能を分析し、それをベースとして、Corebankが有する機能を利用できればそれをそのまま利用（フィット）するが、Corebankが有しない機能（ギャップ）については個別（カスタム）のプログラムを作成することでシステムの開発を進めるという手法「カスタマイズ・ベース・アプローチ」が採用されていました。

④ 要件定義前の「最終合意」

ところが、平成17年5月末に締結予定であった最終合意について、開発範囲・機能の合意に時間を要することなどを理由に、締結が6月末に延期され、さらに平成17年9月末に再度延期されました。

このような度重なる延期を経て、要件定義した成果物が納入される前にスルガ銀行は、同年9月30日、日本IBMとの間で最終合意書を交わすことになります。最終合意書には、作業範囲

1　米IBMが開発した勘定系基盤を日本IBMが日本向けにローカライズしたパッケージシステム。

や価格などを定めた複数の個別契約を締結することを条件に、支払総額が約89億円になることなどが記載されていました。

最終合意以降

① 変更提案

平成17年10月5日に開催された第12回ステアリング・コミッティにおいて、ようやく2回目の要件定義の一部が完了しました。

ところがこの辺りから思わしくない状況が続きます。日本IBMは、平成17年12月12日に、スルガ銀行に対し、開発手法及びスコープの見直しが必要であるなどとして、開発方法及び開発内容の修正を提案したのです。

平成18年3月から、パッケージソフトであるCorebankが有する機能をベースとして、スルガ銀行の業務プロセスを可能な限りCorebankに合わせて変革し、カスタマイズは必要最小限に抑える手法「パッケージ・ベース・アプローチ」が採られることになったのです。これはスルガ銀行の業務に合わせてパッケージソフトをカスタマイズするという開発手法である「カスタマイズ・ベース・アプローチ」とは大きく異なり、スルガ銀行の業務そのものをパッケージの想定する業務形態へ変更するという意味であり、開発手法そのものが大きく異なるものです。

② 開発中断

平成18年4月5日に開催された第17回ステアリング・コミッティにおいて、大部分の設計工程が一応完了し、実装工程（プログラミング工程）を開始することが承認されました。しかし、その後、システム開発は中断することになりました。日本IBMの役員は「パッケージの選択ミスといった一つの原因だけではありません。ただ、準備が不十分だったことは認識しています。またプロジェクト当初に開発手法の選択を誤ったのは事実です」と述べるなど、スルガ銀行の指摘に対して弁明することを余儀なくされていきます。

スルガ銀行と日本IBMは、平成18年11月13日に開催された第23回ステアリング・コミッティで、平成20年5月に本件システムの一部のサービスを開始し平成20年12月末までにシステムを全面稼働させるという基本的な方向性について合意しました。

③ 追加費用要求

ところが日本IBMは、平成18年11月以降、スルガ銀行に対し、開発スコープの削減と開発費用の追加負担に関する提案を行いました。同年12月11日、当初の計画どおりの開発スコープを実現するためには、追加費用として127億円が必要になることを示した上、開発スコープの削減及びそのスコープを前提とした追加費用34億円の負担を提案したのです。しかし平成18年12月22日には追加負担費用額を20億円に減少させた案を提出しました。

④ 再度の変更提案

その後、スルガ銀行と日本IBMの間で、開発スコープや追加の開発費用について合意ができない状態が続きましたが、日本IBMは、平成19年3月19日、スルガ銀行に対し、平成22年1月までに5段階に分けて全面稼動させるスケジュールを提案しました。そして平成19年4月18日にはスルガ銀行に対し、Corebankに代えて、Temenos社の基幹系パッケージソフトである「TCB」を採用する代替案を提案したのです。この提案はスルガ銀行にとっては予期できないもので、スルガ銀行の責任者を激怒させ、スルガ銀行の日本IBMに対する信頼関係は急激に薄れていきます。

⑤ 決裂

スルガ銀行は平成19年5月9日、日本IBMに対し、本件プロジェクトを一旦白紙に戻す旨を記載した書面を交付しました。その後もスルガ銀行と日本IBMは、本件プロジェクトに関する話し合いを続けましたが、結局合意に達することはなく、スルガ銀行は同年7月18日、同月17日付け内容証明郵便を送付、日本IBMに対し、本件最終合意及び本件個別契約を債務不履行(履行不能）により解除する旨の意思表示を行ったのです。

訴訟の結末

上記のような経過で頓挫したスルガ銀行のシステム開発は、当然ながら民事訴訟に発展しました。

スルガ銀行は日本IBMに対してプロジェクトマネジメント義務違反の不法行為に基づく損害賠償請求を、日本IBMはスルガ銀行に対して反訴で個別契約の未払い代金とスルガ銀行の協力義務違反を理由とする投資費用を損害として請求しました。

訴訟の経過は以下の表のとおりです。

表　訴訟の経緯

東京地裁 平成24年3月29日判決	スルガ銀行勝訴 IBMに約74億1366円の支払命令
東京高裁 平成25年9月26日判決	スルガ銀行勝訴 IBMに約41億7210円の支払命令
最高裁 平成27年7月8日決定	双方の上告を棄却 →東京高裁の判決が確定

第1審の東京地方裁判所は、システムの企画段階から要件定義、開発中止に至るまでにスルガ銀行が日本IBMに支払った費用を損害として認定し、日本IBMの反訴請求を全部棄却しました。

これに対して、第2審の東京高等裁判所は、スルガ銀行と日

本IBMが最終合意書を交わした平成17年9月末以降（最終合意以降）の費用に限定して、損害を認定しました。結局、日本IBMの反訴請求はここでも全部棄却されたので、ユーザーであるスルガ銀行が勝訴したことになります。

スルガ銀行と日本IBMは、ともに最高裁判所に上告しましたが、最高裁判所は双方の上告を棄却しました。これにより、東京高裁のスルガ銀行勝訴の判決が確定しました。

実際、私は、裁判所でこの事件の記録を閲覧しましたが、記録は全部で72冊もあり膨大なものでした。システム開発訴訟では、証拠書類が多いので、通常の民事訴訟よりも記録は多くなりがちですが、それにしても膨大な量です。実際に記録を見てみると、金額交渉の進展を示す証拠がなく、宴会の席で日本IBMの責任者が、「箸袋」に数字を手書きして渡したという主張があり、その数字が記載された「箸袋」が証拠として提出され、スルガ銀行との金額の合意を主張する場面もあり、日本IBMの苦しい戦いを垣間見ることもできました。

システム開発　失敗の原因は何か

第1審の東京地裁は、提案から要件定義段階までを詳細に分析し、後半でカスタマイズベースからパッケージベースに、さらにはパッケージ自体の変更まで提案せざるを得なかった事態をふまえて、すでに最初の段階で、パッケージの検討、現行業務との整合性についての検討が不十分であったとして、当初からの日本IBMの責任、74億円余りの支払いを認めていました。

これに対し控訴審の東京高裁は最終合意前には問題はない、最終合意以後に無理なら無理というべきだったとして最終合意以後に限定し、日本IBMの責任を41億円の範囲に限定し認めました。

ベンダーの分析検討不足については一審判決の指摘が正確だと思いますが、ユーザー側であるスルガ銀行にしても日本IBMに迎合していた部分もあり、控訴審判決は最終合意まではユーザーのぐらつきにも責任があると見て、それまでは引き分け、ただ終盤の日本IBMのぐらつきは許しがたいと判断し、喧嘩両成敗的な判断に落ち着いたように見えます。

この両判決をふまえて、ベンダー側の失敗の原因と、ユーザー側の失敗の原因をそれぞれ解明すべきでしょう。

企画・提案段階で何をすべきか

①　ベンダーが行うべきこと

ポイント①

企画・提案段階では、提案するシステムの機能を検証し、想定されるリスクをユーザーに説明しなくてはいけない

東京高裁は企画・提案段階の日本IBMの不法行為責任は、結論として否定しましたが、それでも次のように述べました。

「企画・提案段階においては、プロジェクトの目標の設定、開発費用、開発スコープ及び開発期間の組立て・見込みなど、プロジェクト構想と実現可能性に関わる事項の大枠が定められ、また、それに従って、プロジェクトに伴うリスクも決定づけられる」（中略）

「そうすると、ベンダとしては、企画・提案段階においても、自ら提案するシステムの機能、ユーザーのニーズに対する充足度、システムの開発手法、受注後の開発体制等を検討・検証し、そこから想定されるリスクについて、ユーザーに説明する義務があるというべきである」

としましたが、日本IBMがはたしてこの義務を尽くしていたかについては疑問が残ります。

日本IBMが、後半に入り、開発手法を変更したり、当初の予算をはるかに上回る巨額の追加費用を要求した上で、ついには開発に使うパッケージソフトも変えざるを得なかったという経緯からすれば、スルガ銀行側が十分に事前検証がなされていないと激怒するのも当然でしょう。この点について、東京高裁は「重要プロジェクトを担うベンダとして不見識、無責任とのそしりを受けてもやむを得ない」としました。

第1審判決では、日本IBMが当初の「カスタマイズ・ベース・アプローチ」から「パッケージ・ベース・アプローチ」に変説したことなどを挙げ、「（日本IBMは）本件システム開発を開始するに当たり、Corebankの機能や充足度、その適切な開発方法等についてあらかじめ十分に検証又は検討したものとはいえないし、また、本件システム開発を遂行するに際し、適切な開発方法を採用したものということもできない」と、日本IBMの責任を全面的に認めたのです。

いずれにしても、ベンダーは、企画・提案段階では、提案するシステムの機能を十分に検証し、想定されるリスクをユーザーにしっかり説明しなくてはいけない、ということになります。

② ユーザーが行うべきこと

ポイント②
ユーザーは、自身の業務への影響を慎重に吟味して、現状と変更後の違いを明確にしたうえで、それが受け入れられるかを検討しなければならない

ユーザーは、自身が専門とする業務を知り尽くしているわけですから、その立場から提案内容を厳しく、批判的に検討しなければなりません。パッケージの提案がなされていれば、そのパッケージを使わせてもらい、それで業務が可能なのかを見極めなければなりません。

スルガ銀行は、この時点で自社にシステム部門がありながら、慎重で厳しい吟味をすることなく、トップのいわば独断的で前のめりな政策判断を優先し、日本IBMの提案を鵜呑みにしていたように見えます。旧態依然としたシステムを全面的に否定するのはいいのですが、そこで綿密に進められ、ある意味完成されたスルガ銀行独自の業務分掌、ルールや手順などがあるはずで、それをどの程度変更するのか、変更できるのか、変更しない部分は何かなどを、ユーザー目線で慎重に検討すべきだったのです。

要件定義段階で何をすべきか

① ベンダーが行うべきこと

ポイント③
契約締結後もユーザーに説明し、了解を得ながら、適宜修正・調整を行いながらプロジェクトを進めなくてはいけない

提案内容をユーザーが納得して合意した場合には、基本契約など方向性を決める契約が締結され、実務作業（要件定義）が開始されることになります。ベンダーは契約合意に従って、適切なリーダーシップを取る必要があります。

ベンダーは、「適宜得られた情報を集約・分析して、ベンダとして通常求められる専門的知見を用いてシステム構築を進め、ユーザーである被控訴人に必要な説明を行い、その了解を得ながら、適宜必要とされる修正、調整等を行いつつ、本件システム完成に向けた作業を行うこと（プロジェクト・マネジメント）を適切に行うべき義務を負うものというべきである」（控訴審判決）と述べられています。

要件定義は上流工程と言われ、それ以後を統括する最も重要な骨格を決める作業となります。したがって、ベンダーとしての専門性が如何なく発揮され、またユーザーの業務詳細をベンダーが正確に理解し、適切なシステムを構築するための計画を立てることになります。

この過程が、システム開発ではもっとも大切で、ユーザーがもっとも発言しなければならない部分です。ベンダーはユーザーの説明を丁寧に聞いて、現在の業務を理解し、将来のあるべき姿を示し、修正し、ユーザーの望むあるべき姿を明確にしていく作業になります。

② ユーザーが行うべきこと

ポイント④
ユーザーは、ベンダーが業務の素人であることを前提に、徹底的に現在の業務を理解させ、さらにその課題を摘示して、新システムでどう変わるのか、可能なのかを徹底して議論すること

すでに述べたように、ユーザーは自ら求めるものを作ってもらうのですから、「望むもの」を明確にし、詳細に説明し、希望を形にするように求め続けることになります。

要件定義は、システムを構築するための基本となります。最も重要な設計図なのです。

この設計図には、ユーザーの必要とするものがすべて盛り込まれていなければなりません。後になって、「要件定義に記載がないことは、やれません」と冷たく断られることになりますので、ユーザーとしては望む姿がしっかり入っていることを確認しなければなりません。

スルガ銀行は、要件定義が終了する前に、最終合意を行っています。

これは全く本末転倒で、失敗の最大の原因ともいえます。要件定義が一次と二次となっていることもおかしいことですが、大きなシステムですからそれはやむを得ないとしても、要件定義が終了していない段階、すなわちカスタマイズ・ベース・アプローチで進むという方針にぐらつきがでている段階で、最終合意なるものを締結したわけです。

その後にパッケージ・ベース・アプローチに変更となるわけですから、最終段階で変更するなど、要件定義が十分に行われていないことを示すものという他ありません。

ユーザーとしても、要件定義段階で、厳しく変更部分の検討を行い、適切な判断を行うべきであったことは否定できません。

要件定義後、最終合意その後（ベンダーが行うべきこと）

ポイント⑤
状況の変化があったら、ユーザーのメリットやリスクを考慮した開発変更や開発中止について説明しなければならない

日本IBM訴訟では、すでにこの段階ではなすすべもない状態であったように思いますが、控訴審判決は「ベンダとしては、そのような局面に応じて、ユーザーのシステム開発に伴うメリッ

ト、リスク等を考慮し、適時適切に、開発状況の分析、開発計画の変更の要否とその内容、更には開発計画の中止の要否とその影響等についても説明することが求められ、そのような説明義務を負うものというべきである。」と述べています。

そして、結果的に東京高裁は、この説明義務を日本IBMは十分果たしていなかったことを理由に、日本IBMの責任を認定しています。

それまでの要件定義が一応完成したとするならば、ベンダーとしては、その実現性やリスクを説明すべきであるのは当然として、方向転換したり、提案内容、要件定義を変更するのであれば、その理由と対応を明確に説明する必要があります。

控訴審判決は「ベンダとしての知識・経験、本件システムに関する状況の分析等に基づき、開発費用、開発スコープ及び開発期間のいずれか、あるいはその全部を抜本的に見直す必要があることについて説明し、適切な見直しを行わなければ、本件システム開発を進めることができないこと、その結果、従来の投入費用、更には今後の費用が無駄になることがあることを具体的に説明し、ユーザーである被控訴人（※スルガ銀行）の適切な判断を促す義務があったというべきである。」と述べています（※は著者挿入）。

こうした判断もあろうかとは思いますが、しかし、抜本的見直しが、仕様確定後のユーザーからの一方的な仕様変更要求にあったのであれば、その点を明確にして、その実現方法を説明する義務、変更可能性について、あるいは実現可能性等について説明すべきであるのは当然です。

しかし本件では、すでに結論が見えていたように思います。

スルガ銀行の専務は、裁判に提出された陳述書の中で、日本IBMは「手のひらを返したように」開発費用や開発期間について最終合意の内容を大幅に変更するような提案をしてきた旨を述べていました。

システム開発における記録管理の重要性

この訴訟では、第1審・第2審を通じて、ステアリング・コミッティの議事録に基づいた事実認定が多くなされています。

日本IBMは、裁判の中で、議事録の記載内容はスルガ銀行から修正を加えられたものであり、開発作業の実態を必ずしも反映していないと主張しましたが、東京高裁は、「確定した議事録は、ステアリング・コミッティの作業実態を反映するものとして取り扱うのが相当である」と述べ、日本IBMの主張を退けています。

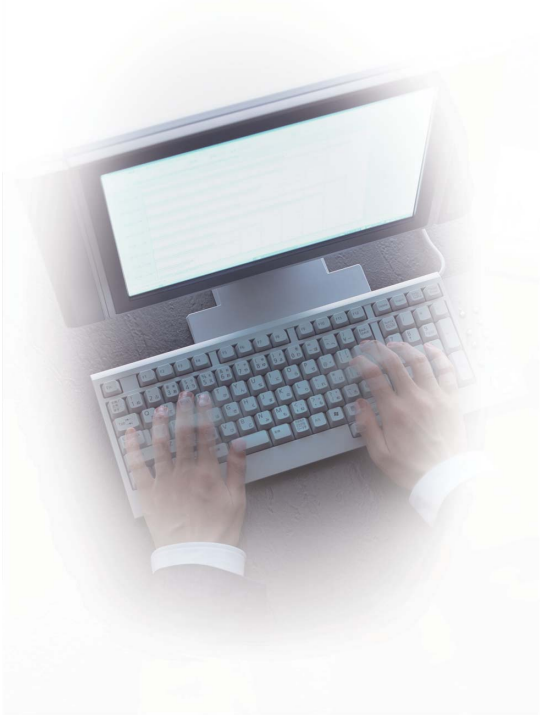
ステアリング・コミッティで議論された要点については、会議の翌々営業日の午前中までに日本IBMが議事録を作成し、議事録データベースに登録し、この議事録で会議の最終的な決定事項を記録化することとされていました。東京高裁は、こうした事情から、日本IBMは議事録作成の意義と方法を当然熟知していたとし、特段の事情が認められない限り、開発作業の内容については、議事録に記載されたとおりのものであると認定するのが相当であると判断したのです。

本連載の第1回目でも記録管理の重要性について述べましたが、ここでも文書記録が、システム開発訴訟で絶大な威力を発揮したことになります。さらにいうと議事録は、特段の事情が認められない限り、そこに記載された通りの議論や言質があったことになりますから、議事録を承認する際には、それが正確に記載されているのか、ユーザーもベンダーも内容をよくチェックする必要があるのです。

まとめ

「スルガ銀行・日本IBM事件」は、システム開発が頓挫したことから、ベンダーが巨額の代償を負うことになった事案として紹介し、これを例題として「失敗例」から見えるベンダーが注意すべきポイントとユーザーが自らを守るポイントを解説しました。

次回の連載第3回目では、成功を裏付けるための上流工程管理を、さらに第4回として成功を裏付けるための下流行程管理について検討したいと思います。



第3回

システム開発成功の鍵 その1—上流工程編

牧野総合法律事務所弁護士法人（JIIMA顧問弁護士） 牧野 二郎（まきの じろう）
牧野総合法律事務所弁護士法人（弁護士） 森 悟史（もり さとし）

前回は、スルガ銀行・日本IBMシステムトラブル事件を多角的に検証して、失敗の原因を探りました。その中で、ユーザー（発注者）側のミス、ベンダー（受注者）側のミスが、重なり合って、結果としてシステム開発が頓挫することを見てきました。

これを基礎にして、今回は、成功させるためのシステム開発の工程管理について検討します。システム開発には多くの人が関係し、多くの作業が発生します。そのためシステム開発には、独特の進め方があります。現在、複数の開発モデルが確立されていますが、ここでは一般的なウォーターフォールモデルを取り上げ、検討してみます。

ウォーターフォールモデルとは

これは名前の通り、滝が上から下に流れ落ちるように、上流の工程から下流の工程に順番に作業を進めていく方法で、原則として、後ろの工程から前の工程に後戻りしないものです。システム開発における工程には、主に上流工程として、企画・要件定義・外部設計があり、下流工程として、内部設計・プログラミング・テスト・データ移行・稼動があります。

各段階での合意形成を基礎にして、段階ごとに管理することができることからウォーターフォールモデルが多く採用されていますが、前の工程に後戻りしないためには当該工程を確実にやり遂げてから次の工程に進むことが必要です。しかしながら、実際には、十分な合意形成ができず、管理が不十分なため作業の手戻りが多く発生し、システム開発が遅延したり、頓挫してしまうことも少なくありません。

今回は、まず上流工程と言われる企画・要件定義・外部設計について検討します。

一般に、ベンダーはシステム開発プロジェクトの経験がある一方で、ユーザー側にはその経験がない場合が多く、システム開発における知識が圧倒的に不足しているのはユーザー側といえます。したがって、ユーザー側が注意すべき点に留意することにします。

実務的にはパッケージソフトを利用したシステム開発、再開発が非常に多くなっていますが、今回は基本的なフルスクラッチ型開発[1]を中心に説明し、パッケージソフトを利用した開発については必要に応じて触れることにします。

上流行程の最重要ポイント その1　企画段階

システム化構想

新たなシステムを導入しようとしている利用者（ユーザー）は、なぜ新たなシステムが必要なのかを明らかにする必要があります。システムを開発するということは、現状の業務に問題や課題があり、それを新たなシステムを導入することで解決したいからでしょう。すなわち、経営課題を解決するためにシステムを利用するのです。そのためシステム開発は経営戦略と整合したものでなければなりません。そこでユーザーとしては、現行システムの調査・分析を行い、システムに求める「課題」を抽出し、システム化における基本的な方向性として「課題の解決方向」を検討します。これをシステム化構想と言います。もっとも、この段階におけるユーザーの要求事項は概要にすぎず、次の要件定義工程で、その詳細を具体化することになります。

システム化計画

次に、システム化をどのように実現するのかというシステム化計画が必要です。具体的には、ユーザーの業務を分析し、どのような情報システムが必要となるのか、どのように開発プロジェクトを進めるかなどの基本方針を策定します。

以上のように、企画の工程（フェーズ）では、ユーザーは、システム化構想とシステム化計画を明らかにして、自らの要求内容を明確にする必要があります。方向性をしっかりと示さないと、何のためにコストをかけてシステム開発するのかわからなくなり

1　既存のものを流用せずに一から作り上げること。

ます。また、計画をしっかり立てないと、システム開発におけるコスト、スケジュール、品質を守れなくなります。システム開発は多くの費用と人員を割いて行われるものです。したがって、しっかりとした方向性と計画が必要なのです。

FIT&GAPコンサル（企画段階FIT&GAP）

パッケージソフトウェアを利用する方針の場合、企画工程においてFIT&GAP分析を行うことも必要です。

FIT&GAP分析とは、ユーザーの現行業務やユーザーの要求事項と、使いたい、使えると思われるパッケージソフトウェアの内容を比較し、パッケージソフトウェアを利用してユーザーの希望する業務を実現することができるかどうかを検討する作業です。このFIT&GAP分析は、後述の要件定義工程で本格的に行うのですが、企画工程におけるFIT&GAP分析は、パッケージソフトウェアを吟味して、プロジェクト計画立案と概算見積り算出のために行われます。

企画段階でのFIT&GAP分析については「システム再構築を成功に導くユーザガイド 第2版 ～ユーザとベンダで共有する再構築のリスクと対策～」[2]でも成功の鍵として指摘されているところです。もっとも、ユーザー企業ではFIT&GAP分析を行える人間は少なく、ユーザー自身が行うのは現実的ではありません。

そこで、ベンダーの中には「フィッティングコンサル」というサービスを提供しているものがあります。これは、パッケージソフトが使えるかどうかに焦点を当てて、現行業務に基づいたユーザーの要求事項の概要を固め、この要求事項にフィットするパッケージソフトウェアを吟味し、システム開発の方針（パッケージベースとするか、フルスクラッチ方式で進めるかの判断）等を定めるものです。

なお、システム化構想やシステム化計画に記載されている内容は、企画段階での決定であり、開発のための契約などがない段階ですから、直ちに開発対象や開発内容となるわけではありません。開発対象となるためには、開発請負契約書に明確に記載されるか、次に説明する要件定義書や外部設計書に明記され、合意形成されていることが必要です。東京地判平成22年12月28日でも「本件プロジェクト計画書は、被告の説明を基に原告が作成したものであると推認されるものの、このことから直ちに、本件目的の達成が原・被告間の契約の内容になっていたものと認めることはできない」と判示しているように、仮にベンダーの説明を受けて作成されたプロジェクト計画書に記載されていた内容であっても、その内容が直ちに開発内容とはならないのです。

2　独立行政法人情報処理推進機構技術本部ソフトウェア高信頼化センター・平成30年・29頁参照

上流工程の最重要ポイント その2　要件定義

要件定義とは

①　要件定義の役割

要件定義とは、ユーザーがシステムに要求する機能を定義する作業のことです。ユーザーが求めるものが明確でないと、開発業者（ベンダー）は何を作ってよいのかわかりません。また、ユーザー自身も、自分が本当に必要なものが何なのか、実はよくわかっていないことがあります。そこで、システムに求めるものを明らかにし、要件定義書を作成します。ここで、要件定義は必ず書面化し、明確に言語化する必要があります。ポンチ絵だけでは、概念も位置関係も曖昧になるため、人によって考え方、とらえ方が異なってしまい、ユーザーが作って欲しいシステムとベンダーが作ろうとするシステムに相違が生じます。

ベンダーは、この要件定義書をもとに設計書を作成し、システムを作っていくことになりますので、要件定義書は、システム開発においてその基盤となる重要なものです。

②　要件定義を作る主役とは

要件定義は、ユーザーの要求事項をまとめるものですから、本来ユーザーが行うべきものです。しかしながら、システムに精通しておらず、また、他のパッケージなどを見たこともないユーザーが自ら要件定義を行うのは無理というべきです。

そこで、現在ではベンダーが、ユーザーが行う要件定義を支援し、ユーザーの要求を聞き取り、取りまとめ、要件定義書という書面にまとめる作業を行うことが多いです。

ただ、あくまでも要件定義はユーザーの要求の整理であり、ユーザーが行うものであり、ベンダーはそれを支援する立場にある以上、ユーザーはベンダーに対して、要件定義を委任する「委任契約」という事になります。この点、東京地判平成22年7月22日でも「開発の出発点である要件定義を確定する工程については、注文者の要求をまとめる工程であると定義されるとおり、注文者側の意向によってその内容が決せられることになるのであるから、注文者側がどのような内容のソフトウェアの開発を望んでいるかを提示又は説明する責任は、注文者側にそのような能力がないことが前提になっているなどの事情がない限り、注文者側にあるというべきである」と判示しています。

この判例の立場でも、ユーザー側ができないときは、ベンダーに委任すべきであり、委任契約によりベンダーは要件定義を完成させるために受任業務を誠実に実施する義務を負うこと

になります。

要件定義書の記載内容

要件定義には「業務要件定義」と「システム要件定義」があり、さらに「システム要件定義」には「機能要件」と「非機能要件」があります。

まず業務要件定義ですが、業務要件とは、システム化の対象となる業務の流れを把握し、そこから業務における必須機能、重要機能を明確化することです。現状の業務を洗い出し、どこに問題・課題があるのかを明確にした上で、明らかになった問題・課題を解決すべく、新たな業務フローを検討することになります。その際には、これまでの業務フローを単純化し、効率的な業務フローにすることも重要です。人に依存した複雑なフローでは、システム化の際にも多くの困難を引き起こします。可能な限り単純化し、整理すべきでしょう。

さらに作成された新たな業務フローをもとに、どの業務をシステム化し、どの業務を手作業のまま残すのかを明らかにします。ルーティンの業務はシステム化しやすいですが、他方で、例外的な取扱いが複数含まれている部分は、手作業のまま残すなどの選択が重要です。

次に、システム要件定義が検討されることになります。システム要件定義では、ユーザーが求めるシステムにおいて実現する機能を明確にすることです。人手で対応する部分を除いて、システム化する範囲を決め、そのシステム化の範囲で必要な機能を洗い出します。この機能を整理すること、言語として明確に定義すること、すなわち要件として具体的に明確にするのが要件定義であり、最重要課題なのです。この機能の明確化、要件化が曖昧であると、開発プロジェクトそのものにも影響します。

要件定義を画面説明など、ポンチ絵で済ませるベンダーが多数いるようですが、要件定義の意味を無視するものであり、トラブルの原因を作っているようなものですので、こうしたベンダーには絶対に依頼しないようにしてください。

最後にシステム要件として明確にする機能要件の他に、非機能要件があります。これは、機能要件以外で必要となる重要なポイントを示すものです。非機能要件には、性能（処理速度や同時処理可能な件数など）・拡張性（将来予定される拡張に対応できること）、品質、運用・保守性、移行性、セキュリティなどがあり、システムの「質」についての要求事項を明確にすることを言います。システムは、ユーザーが求める機能が備わっていれば足りるのではなく、使いやすく（操作が簡単で、明瞭であるなど）、保守しやすく、トラブルが少ないものであることが必須です。ベンダーの中には、非機能要件の重要性を理解せず、非機能要件・質にかかる重要な条件がほとんど記載されていない要件定義書を作るものがいますが、導入後、現場のクレームが続出し、使い物にならないとしてトラブルになる可能性が高いので注意が必要です。

上流工程の最重要ポイント その3　外部設計

外部設計（基本設計）とは、ユーザーから見たシステムの振る舞い、システムの方式、操作する画面のレイアウトや操作方法、帳票の種類・内容、データベース構造、データ項目などを決定し、外部設計書を作成する作業です。要件定義がユーザーの希望をまとめるものであるのに対し、外部設計はユーザーの希望を実現する方法をユーザーに提示するものであって、システムが表示する画面や作成できる帳票にとどまらず、システムの方式（システムアーキテクチャ）を明確にする必要があります。

この外部設計は、ユーザーの意向・希望を実現するための姿・動きとして、機能の実現の姿（仕様）を確認できる最後の場であることから、ユーザーは、ベンダーから提出された外部設計書によって、自身の要求が実現できているかどうかを丹念に確認する必要があります。ベンダーは、ユーザーの承認を得るため、外部設計書はユーザーが理解できるようにわかりやすく作成し、詳細に説明する必要があります。すなわち、画面のイメージだけでは、ユーザーは当該画面にはどのような機能があり、どのように使用するのか、どのボタンをクリックするとどの画面に遷移するのかなどが理解できないことから、画面イメージのみならず言語による説明、データ遷移の流れ、インプットとアウトプットの関係などを言葉で明確に説明する必要があります。

ところが、ベンダーの中には、外部設計（基本設計）と称して、画面キャプチャーだけを示して、済ませるものがいます。ユーザーとしては、自分の要求が実現できるかを審査する最後の段階ですから、外部設計書の内容、すなわち画面キャプチャーだけではなく、入力画面からの動き、データ遷移、アウトプット画面への流れ、実際のアウトプット、データベース、データ項目などの基本構造についてもしっかり確認すべきです。

要件定義・外部設計工程での注意事項

以上のように要件定義及び外部設計は重要な工程なのですが、専門的な内容を含んだものであり、多くのユーザーにとってわかりにくいものです。そのためユーザーは、分からないところは勝

手に想像して、良い方向に理解しようとします。ベンダーは明確に書いていない部分は「一切やらない」という姿勢でいますので、ここに大きな認識の乖離が生まれるのです。

システム開発が失敗する大きな原因は、この要件定義や外部設計がユーザーの理解できる正確な日本語で表示されておらず、ユーザーとベンダーの考えが一致していないことにあります。したがって、要件定義や外部設計の作業は慎重に行う必要があります。

要件定義や外部設計を進めるにあたっては、以下の点に注意する必要があります。

- **要望事項を必ず要件定義書や外部設計書に具体的に明記してもらうこと**

要件定義工程、外部設計工程において、ベンダーがユーザーの要望を聞いて、要件定義書、外部設計書を作成します。そしてユーザーがこの要件定義書と外部設計書を承認すると、これらをもとにベンダーが開発作業を行います。そのため、要件定義書、外部設計書に記載されていない機能・仕様は、原則として開発対象外となります。したがって、ユーザーの要望事項は必ず要件定義書、外部設計書に具体的に明記してもらう必要があります。

東京地判平成24年2月29日も「契約書を含め、本件契約締結の前後に作成された本件システムの仕様に関連する書面には、本件システムに上記機能を盛り込むことを要するとする記載は見当たらない」、証人は換金機能の重要性を供述するが、そうであれば「確実に書面等をもって確認するのが当然であるのに、同証人自身は、営業活動を担当していたというのであって、どの時点で原告の誰に対して換金機能を実装するよう確認したというのか明らかでなく、換金機能の実現に関する法規制上の障壁について当時検討していたかどうかも明らかでない」として、「本件システムに電子ポイント換金機能を盛り込むことが本件契約の内容となっていたと認めることはできない」と判示し、仕様書（設計書）に記載されていない機能は契約の対象外であるとしています。

- **議事録等を残しておくこと**

要件定義工程では、ベンダーはユーザーの要求事項を明らかにする作業を行います。そのためベンダーは、ユーザーから書類をもらって検討したり、ユーザー（特にシステムの実際の利用者）に対しヒアリングを実施します。そもそもユーザーはシステム開発において何が重要なのかわからず、またベンダーはユーザーの業務をほとんど知りません。そこでベンダー側は、必要な書類をピックアップし、ユーザーに指示するとともに、ヒアリングにおいても自ら積極的に必要な情報を聞き出す工夫が必要となります。

ヒアリングの際には必ず議事録を作成してください。トラブルとなった事例においては、要件定義書には記載がないが、ヒアリングの際にはユーザー側からベンダー側に説明していたと、ユーザー側が主張できるケースが多数見受けられます。ユーザーは、希望を明確に伝え、かつ、そのことを議事録に記載し、両者で確認しておく必要があります。

なお、東京地判平成16年6月23日は、「本件ソフト開発契約の仕様書には、遠隔操作機能について記載されていない。しかし、もともとソフトウェアの仕様書は複雑なものであり、専門家でなければ容易にわかり得ないものであるから、仕様書に記載がないからといって、契約の内容になっていないということはできない」としたうえで、「遠隔操作機能については、本件ソフト開発契約の仕様書には記載がないものの、原告は、被告から口頭で、遠隔操作機能の開発依頼も受けたものというべきである」と判示して、仕様書（設計書）に記載がない場合であっても、開発範囲であることを認定しました。この事案では、打ち合わせの過程で「遠隔操作機能」についての話が出ていたことや、ユーザー作成のいくつかの書面に「遠隔操作機能」を示す記載がありました。裁判所は、このような事情を斟酌して口頭での合意を認めました。

このように仕様書（設計書）などの記載がない場合でも、議事録などの書面から開発範囲が認定されることもあるので、議事録は是非とも残しておきましょう。

もっとも、前述の東京地判平成24年2月29日のように、基本的には仕様書（設計書）の記載内容で判断されます。したがって、まずは仕様書（設計書）や要件定義書に記載されているかどうかを慎重に確認することが重要で、同時に議事録の作成により、正確な記録と管理を実施するということになります。

- **パッケージソフト利用の有無を検討すること**

現在では既存のパッケージソフトを利用し、足りない部分だけカスタマイズ、アドオンするシステム開発が増えています。パッケージソフトを利用する場合、要件定義段階で、業務をパッケージの機能に寄せるか、現状の業務に合わせてシステムをカスタマイズするかを決める必要があります。パッケージ機能に寄せるとは、パッケージに合わせて、ユーザーの現在の業務フローを変更することを意味します。そのため、パッケージに合わせた新しい業務フローによる業務遂行が可能かどうか、効率的かどうかを確認する必要があります。

ベンダーは、要件定義工程において、パッケージソフトの機

能とユーザーの現状の業務のFIT部分（業務とシステムが一致している部分）とGAP部分（システム、パッケージにはない機能）を分析するFIT&GAP分析を行い、その結果をユーザーに示し、ユーザーにどの部分をパッケージに寄せ、どの部分をカスタマイズするかを決めてもらう必要があります。FIT&GAP分析の結果、GAP部分が大きい場合は、パッケージソフトを利用しないことも検討する必要があります。

前述の東京地判平成22年12月28日においても「パッケージソフトウェアは、ソフトウェア会社において一般のユーザーが標準的に必要とする業務処理方法をプログラム化したものであり、ユーザーがこの処理方法をベースとして導入することにより、コンピュータ・プログラムのすべてを独自に自前で開発するカスタムソフトウェアの場合に比べ、効率的にシステムを完成でき、システム完成後の基幹ソフトウェアの保守及び改良についてもパッケージソフトウェア会社から効率的なサービスを受けることができるというメリットがある。したがって、パッケージソフトウェアを利用した情報システム開発を円滑に進めるためには、パッケージソフトウェアの導入を支援するベンダーと社内業務処理に精通したユーザーが共同で、パッケージソフトウェアの標準的な業務処理方法とユーザーの社内業務処理方法との差異を明確にし、この差異を解決するために、ソフトウェアの機能を変更、追加すべきか、社内業務処理方法を変更すべきかを、差異毎に検討し、決定することが必要である」と判示し、パッケージの機能とユーザーの現状業務の差異分析を行い、業務を変更すべきかを決定することが必要であると述べています。

- **業務の特殊性について考慮すること**

システム開発を行うには、各業界の特殊性を考慮する必要があります。各業界には特有の業務・フローがあり、他の業界で利用されているシステムやパッケージソフトをそのまま流用することができない場合があります。例えば営業倉庫の場合、通常の在庫管理システムとは異なるフローがあり、通常の在庫管理システムや販売管理システムを流用することは難しいのです。この点はベンダーがよく知らない部分でもあり、安易な判断で使えないパッケージソフトを押しつけてくることがありますので、注意が必要です。そのためにも要件定義段階で、ユーザーの専門とする業界の特殊性、さらにはユーザー特有の工夫や技能を前提とした業務の特殊性、作業の独自性を考慮して、必要な機能、仕様を明らかにしておくことが必要となります。

- **安易に要件定義書や外部設計書を承認しないこと**

要件定義工程が終了する際、ユーザーはベンダーが作成した要件定義書を検収し、承認する必要があります。また外部設計工程が終了する際には、ユーザーは外部設計書を検収し、承認する必要があります。ところが、それまで要件定義書や外部設計書を見たことがないユーザーは、ベンダーがシステム開発の専門家であることから、大丈夫だろうと安易に考え、しっかりとした検収をせずに、要件定義書や外部設計書を承認してしまうことがあります。

要件定義書が承認されると、ベンダーは、これでユーザー要望の機能が出尽くしたとして、外部設計に入ります。また外部設計書が承認されると、ベンダーは内部設計・プログラミングに入ります。したがって、一旦、要件定義書・外部設計書を承認すると、新たな機能・仕様の追加や変更を要求することが難しくなります。

したがって、検収は慎重に行う必要があります。もっとも、ベンダーが作成した要件定義書や外部設計書をユーザー側では理解できず、検収できない場合には、システム開発に精通している第三者（システムの専門家）に要件定義書や外部設計書を見てもらうことが必要です。よく理解せずに承認するのは、トラブルを呼び込むようなものです。

まとめ

以上、システム開発における上流工程の内容やその注意点を検討しました。システム開発の成否はこの上流工程で決まるといっても過言ではありません。ユーザーの希望、課題を明確にして、その課題解決方法を確定するのが上流工程なので、ここで発生した誤解はシステム開発全体の欠陥を固定化するものとなります。

上流工程では、ユーザーの果たす役割が大きくなります。ユーザーが承認しなければ前に進められないからです。安易な承認（検収）は命取りになりますので、慎重に、分かるまで確認して、分からない点は放置したり曖昧にせず、理解するまで議論し、整理し、言語化して進めるようにしてください。

第4回

システム開発成功の鍵 その2―下流工程編

牧野総合法律事務所弁護士法人（JIIMA顧問弁護士） 牧野 二郎
牧野総合法律事務所弁護士法人（弁護士） 森 悟史

前回は上流工程における注意点を検討してみました。今回は、続いて下流工程の内容とその注意点を検討します。

ウォーターフォール型システム開発は、企画、要件定義、外部設計、内部設計、プログラミング、テストという工程をたどりますので、内部設計、プログラミング、テストなどが下流工程となります。下流工程は、主に、要件定義、外部設計書を基本として、その実現を請け負うことになるため、請負契約で行なうことになります。

上流工程がしっかりしていない場合、不具合は下流工程、特にテスト段階で発見されることになります。下流工程段階で仕様が不充分であることが判明した、または仕様変更の必要が生じた、あるいは運用テストしたがバグだらけだったなど多くのトラブルがあります。

今回は下流工程の内容について概説するとともに、特に問題となる変更管理、テストとともに、下請業者を利用した場合の注意点について検討します。

内部設計・プログラミング

内部設計とは、システムの内部の動き、処理の仕方についての設計です。要件定義で確定された機能要件・非機能要件や外部設計で明らかにされた仕様を、どのように実現するのかということを決める工程となります。この作業を詳細設計と呼ぶこともあります。

次に、プログラミングとは、コンピュータに処理させるプログラムを、プログラム言語（プログラミング言語）を使って作成する作業のことです。

これらの内部設計及びプログラミングの作業はベンダーが行いますので、この工程においてユーザーは、進捗管理を行い、スケジュールが守れるように管理することが大切になります。

変更管理

変更管理とは

要件定義、外部設計が承認され、外部仕様が確定した後であっても、実際に作り始めたときになって初めて明確になる課題や定義漏れ、仕様書（設計書）の記載内容の矛盾や不足が発見されることがあります。また、ユーザーとしては、時代の変化に伴い、新たな要望が出てくることもあります。

東京地判平成15年5月8日でも、「システム開発作業においては、作業を進める中で当初想定していない問題が明らかになったり、より良いシステムを求めて仕様が変更されたりすることが普通であり、これらに対応するために追加の費用が発生することはいわば常識であって、追加費用が発生しないソフトウェア開発など稀有であるといって過言ではない」と判示されているほどです。

このように、要件定義、外部設計の検収により仕様が確定した後であっても、仕様（機能追加等も含め）を変更する必要性は否定できません。ただし、仕様の変更を無制限に認めると、システム開発プロジェクトのスケジュール遅延、コスト増大、品質の低下を招く危険性があります。そこで、変更要求に対しては、その変更による手戻りの範囲、全体への影響、スケジュール、費用の増減を正確に測定し、採用の有無を判断する必要が出てきます。それが変更管理となります。

変更管理手続

一旦確定した仕様を変更するには、あらかじめユーザーとベンダーで定めた手続に従い、本当に変更する必要があるのか、どの範囲をどのように変更するのか、スケジュールやコストに影響はないのかなどを検討する必要があります。そのためには、以下の手続をとることが必要です。なお、変更要求の検討にも工数がかかり、費用の増加になることをユーザーは心得ておく必要があります。

① 変更要求

ユーザーからの変更要求が多く発生します。要件定義や外部設計で現在の姿とこれからできる姿が見えてきた段階、すなわちユーザーのシステムの理解が進んだ段階で、新たな要求や疑問が出てくるものです。さらに長期の作業である場合は、着地点を変更すべき事態になることもあります。そこで、五月雨式に出すのではなく、一元化して、誰がどのような方法(例えば書面)で変更を求めることができるのか、変更要求の手続を予め明確にして、ベンダーの検討作業を整理できるようにしておく必要があります。

② 要求内容の検討

変更要求を受け取ったベンダーは、その要求内容を検討します。その際には、どの仕様をどのように変更するのか、仕様を変更した場合に他の仕様に影響がないのか、他の仕様に影響がある場合には、他の仕様をどのように変更するのかなどを検討する必要も出てくることがあります。仕様変更を行う場合には、工数にも影響を及ぼします。そのため、スケジュール、コスト、品質への影響調査も必要となります。

こうした変更要求の検討にはかなりの工数がかかるので、ユーザーとベンダーとで、変更要求検討による費用増加の問題を、話し合っておく必要があります。

根本的な機能の追加や変更が必要となったり、基本構造に影響する場合には、もう一度要件定義からし直すことも考えられます。そのときは、もはや変更管理ではなく、開発作業自体の変更を想定した検討を行い、当事者の緊密な議論、情報共有により、打開策を検討する必要があります。

③ 仕様変更承認

ユーザーとベンダーで話し合いを行い、必要な仕様変更であること、その影響を承認できることを条件として、仕様変更が合意されます。

変更管理における注意点

変更管理手続において、注意すべき点は以下のとおりです。

① 仕様の具体化か、変更かを明確にすること

要件定義、外部設計において、その内容、仕様が詳細に決定されていれば、それに応じて変更となるかが判断できるはずです。

しかし、具体化には程度の差もあり、仕様を詰める中で、意見の相違も出てきます。

したがって、変更内容の検討の際には、当初の予定である要件定義、外部設計の想定範囲がどの範囲であったのかを慎重に議論して、その範囲の中なのか(仕様の具体化の範囲)、それとも仕様変更なのか(元の仕様とは異なる仕様とすること)を区分することが重要です。

② 承認内容の書面化

仕様変更が承認された場合、どのように仕様を変更するのか、またスケジュールやコストがどのように影響を受けたかについて、明確にしておくことが必要です。そこで、仕様変更を承認する場合、承認した内容を必ず書面に残さなければなりません。

また、仕様変更に伴うドキュメント(要件定義書や外部設計書など)の修正も必要です。要件定義書や外部設計書は、この後のテスト工程において、出来上がったシステムを検証する際に利用されます。そのため、変更管理承認の内容を書面にして残すとともに、ドキュメントの修正も併せて行う必要があります。

テスト

テストの種類

テストとは、プログラミング工程終了後において、システムに潜む欠陥を検出する工程です。なお、厳密に言うとテスト作業は、前述のプログラミング工程に含まれるとされていますが、ここでは、便宜上、別の工程として説明します。

テストには主に、単体テスト、結合テスト、総合テスト、受入(運用)テストがあります。単体テストは、個々のモジュール(部品)のみを対象とするテストで、対象のモジュールが仕様書・設計書で要求された機能・仕様を満たしているどうかをテストするものです。結合テストは、単体テスト後に行う複数のモジュールを組み合わせて行うテストで、主にモジュール間のインターフェースがうまく機能するかどうかに注目して行うテストです。総合テストは、システムが全体として予定どおりの機能・仕様を満たしているかのテストで、システムテストとも呼ばれます。総合テストでは、機能要件のみならず、非機能要件についてもテストの対象となり、システムの性能についても確認をする必要があります。

これらのテストは、テスト環境、テストした内容、その結果などが分かるように整理され、後にユーザーに確認を求めるものとなります。そのため、テスト要件を予め当事者で確認しておく必要があります。

次に、受入(運用)テストが行われますが、このテストは、ユーザーが行うもので、ユーザーの担当者(システム利用者)が、実際に業務データを使用するなどして開発されたシステムを稼働させてみて、業務運用上の支障がないかを確認するものです。

一般的には、単体テスト、結合テスト、総合テストはベンダー側で行い、受入（運用）テストはユーザーが中心に行います。ただ、ユーザーはシステムに慣れていないことから、システムの操作説明などをベンダーが行う必要もあります。ベンダーの指導の下で、運用上の問題がないか、確認する作業となります。

受入テストが終わり、ユーザーが承認（システムの納品を受け入れること）すると、システム開発プロジェクトは終了し、その後、移行、保守管理の工程に入ります。

以上、テスト段階において、注意すべき点を検討します。

テストにおける注意点

①　わからないときは第三者の専門家に見てもらうこと

ベンダーが内部で行ったテストの結果が適正なものかは、専門的判断によることも多いことから、ユーザーは、テスト結果がよく分からない場合や、その的確性の判断ができない場合には、専門家に相談する必要があるでしょう。

②　容易に承認しないこと

受入テストは、ユーザー自身が行うものですが、さまざまなテストが必要となります。繁忙期を想定したテスト、間違えて入力してしまった場合の対応のテスト、経営分析のためのデータ取り出しの容易性のテスト、現場従業員に分かり易いか、手順は合理的かなど、多様なテストが必要です。それを十分に行わずに、その後、当初求めていた機能が実装されていないとか、処理速度が遅すぎるなどとクレームを入れても、ベンダーは聞き入れない場合が多いものです。対処するとしても追加料金を要求されることが多いでしょう。

このように、受入テストによる承認はシステム開発プロジェクトにおけるマイルストーン（重要な節目）であり、ユーザーは、容易に承認せずに、慎重に判断する必要があります。

③　非機能要件もチェックすること

テストでは、自分が求めた機能が満たされているのかといった機能要件に目が行きがちですが、テスト段階では、非機能要件（非機能要件の内容については、前回7．8月号を参照）についても確認する必要があります。

非機能要件については、要件定義工程などで明らかにしておくべきで、テスト段階で明確にするものではありません。あくまで、要件定義などにおいて非機能要件を明確に定義し、それをテスト段階で、システムが満たしているのかを確認するのです。東京地判平成23年８月26日でも、「システムの安定性は、機能の有無とは異なる要件（非機能要件）であるとともに相対的な概念であるから、原告に対して本件システムの安定性の確保を法的に義務付けるためには、本件システムの稼働率（障害発生の頻度、発生時間の上限等）等について具体的な数値をもって原・被告間で合意することを要するのが本来というべきである。」と判示しています。

システムの完成

すべてのテストが終了し、ユーザーがテスト結果を承認すると、システムの完成が正式に認められます。もっとも、ユーザーの適正なテストの後の承認があった場合であっても、実際にシステムを使って、一定程度の期間が経過した後になり、はじめて不具合が見つかることがあります。この場合、ユーザーとしては、承認はしたけれども、テストでは発見できなかっただけであり、もともと不具合は存在したのだからシステムは完成していないと考えることもあります。

では、システムはどのような場合に完成したと言えるのでしょうか。

東京地判平成14年４月22日では、「請負人が仕事を完成させたか否かについては、仕事が当初の請負契約で予定していた最後の工程まで終えているか否かを基準として判断すべきであり、注文者は、請負人が仕事の最後の工程まで終え目的物を引き渡したときには、単に、仕事の目的物に瑕疵があるというだけの理由で請負代金の支払を拒むことはできないものと解するのが相当である。」と判示しています。すなわち、瑕疵があるという程度であれば、システムは完成していると判断されることもあるのです。

また、ほんの一部のみが未完成であった場合についても、全体として完成していたと判断されることがあります。東京地判平成17年４月22日は、「被告は、原告の業務が完了していないと争うが、前記認定のとおり、414本のプログラムについては、被告がその製作を完了し、そのうち407本の検収が完了している。原告が自認する未完成の８本については、被告からの仕様の提示がないために設計ができなかったものであり、その未完成部分が全体の分量に比べて少量であることに鑑みても、この点を請負業務の未完成の理由として主張することは信義則上許されない」と判示しています。

以上から、ごく一部の機能・仕様が完成していなかったり、単なる瑕疵（バグ）程度のものであった場合は、システムの完成が認められます。

他方で、システムが完成し納品・検収がされた後においても、瑕疵が重大であり、契約の目的を達成することのできない程度

の瑕疵である場合には、契約を解除し、前払金の返還と損害賠償の支払いが認められることがあります。

以上から、必須の機能が無い場合や、処理能力が業務の遂行に耐えないような場合など、契約の目的を達成することができない程度の重大な瑕疵が存在する場合には、仮にユーザーにおいて検収（承認）してしまったとしても、救済されることがあります。

外注・下請管理

システム開発における外注・下請けの必要性

現在、システム開発において、ベンダーから別の業者への外注・下請けが盛んに行われています。複雑なシステム開発や大きなシステム開発の場合、ベンダー1社のみでは対応できず、複数の外注・下請業者を利用することがあります。また、特殊な機能を開発する必要がある場合にも、別の専門業者に外注・下請けがなされることがあります。そのため、ユーザーとベンダーとの間の契約書においても、外注･下請けが認められているケースがあります。特殊機能の開発の場合、その機能についての専門知識がある専門業者への再下請けであれば、品質も確保できるため、外注・下請け・再下請けは認められるべきでしょう。他方で、単なるベンダーの能力不足、人手不足、単なる名義貸しによる外注・下請けの場合は注意が必要です。これまで全く開発したことのないシステムを、あたかも自分達で開発できるかのようにユーザーに説明し、ユーザーと契約を締結してから、実際に開発できそうな別の下請業者を探すといったベンダーが実際に存在します。このようなケースでは、こうしたベンダーには開発経験がほとんどなく、適切にプロジェクトを管理できず、失敗する可能性が極めて高くなります。

このように、システム開発における外注・下請は、頻繁に行われているものの、その管理を怠れば、システム開発プロジェクトに悪影響を与えることにもなりかねません。

そこで、ベンダー側、ユーザー側双方が、外注・下請管理を行う必要があります。以下では、ユーザー側のみならず、ベンダー側も含めて、どのように外注・下請管理を行うべきかを検討します。

ベンダーにおける外注・下請管理

実際に、他の業者に外注・下請けを出すのはベンダー側であることが圧倒的に多いでしょう。したがって、ベンダーが外注業者・下請業者を管理するのが原則です。ベンダーは、具体的に以下の管理を行う必要があります。

① 発注段階

システム開発プロジェクトを成功させるためには、当該システム開発に適した外注業者・下請業者を利用することが必要です。特殊な機能を含んだシステム開発の場合は、その機能の開発を得意とする業者に発注すべきでしょう。ベンダーは、システム開発の専門家として、他の業者の得意分野等を十分吟味して、適切な業者を選定しなければなりません。

そこで、ベンダーとしては、予めどのようなケースにどのような業者を選定するべきかという選定基準を定めるとともに、その基準に従い、当該システム開発に適した外注業者・下請業者を選定する必要があります。選定基準には、ただ単に、費用だけでなく、同じようなシステム開発を行った経験があるか、適切な人員が確保されているか、どのようなプロジェクト管理を行ってきたのか、資金面での問題は無いか（資金ショートする可能性は無いか）なども含める必要があります。

また、適切な業者を選定するためには、他の業者の得意分野や実績等の動向を把握しておく必要があります。

さらに、外注業者・下請業者との契約の際には、外注業者・下請業者の作業状況を確認・監督できるような規定を盛り込んでおくことも必要です。通常、プログラミング等の作業を外注・下請けに出す場合は、請負契約となります。請負契約の場合、民法上、委任契約（民法645条）とは異なり、報告義務がありません。そのため、契約で定めておかないと、成果物の納入時まで、作業内容を確認できないことになります。そこで、契約において、作業内容・作業状況の確認ができるようにしておくべきです。

② 作業段階

次に、ベンダーは、外注業者・下請業者の作業を確認する必要があります。

ベンダーは、外注業者・下請業者との契約で定めた事項に従い、適切に、外注業者・下請業者の作業内容・作業状況を確認します。特に、スケジュールどおりに作業が進んでいるのかについては、週次、月次で確認する必要があります。外注業者・下請業者の作業内容・作業状況を適時、把握していれば、仮にトラブルが発生したとしても、早めにそのトラブルを認識し、それをリカバーする対策がとれますが、作業内容・作業状況の確認を怠っていると、気付いたときには、納期（スケジュール）には間に合わず、とるべき手段・対策がなくなってしまい、手遅れとなってしまうということもあり得ます。このようにならない

ためにも、ベンダーは常時、外注業者・下請業者の作業内容・作業状況を確認しなければならないのです。

なお、請負契約（委任契約も同じ）の場合、外注業者・下請業者の個々の作業員（プログラマーなど）に対し、ベンダーから直接指示を出すことは、法律上できないとされています（偽装請負問題）。これをふまえて、下請け、外注にかかる契約書、仕様書には、必要な協議の設定、立ち入り検査などの条項も規定し、十分管理を行なう必要があります。

③　納品段階

最後に、ベンダーは、ベンダーが外注し、下請けした事業者から成果物を受け取る際に、納品された成果物が本当に発注した機能・仕様を満たしているのか、品質が確保されているのかなどを、ユーザーの目線で確認する必要があります。これは、ベンダーによる単体テスト、結合テスト、総合テストの先出しとも言えます。

ユーザーが受入テストを行うように、ベンダーは、外注業者・下請業者からの納品物を慎重に確認する必要があります。特に、ベンダーは、ユーザーと異なり、システムの専門家なのですから、専門家の目を通した確認をしなければならないのです。

ユーザーにおける外注・下請管理

以上のように、一般的には、ユーザーではなくベンダーが、外注業者・下請業者を管理・監督しますので、ユーザーが外注業者・下請業者を直接監督したりすることはないでしょう。もっとも、どのような外注業者・下請業者を利用しているのかは、開発しているシステムの出来、不出来にもかかわる上、スケジュールが遵守できるのかにも影響しますので、ユーザーにとっても無関係とは言えません。

そこで、ユーザーとベンダーとの間の契約書において、ベンダーが外注・下請業者を利用しようとする場合には、事前にユーザーの承諾を得なければならないという規定を入れた上で、実際に外注業者・下請業者を利用しようとする際には、ベンダーに説明を求めるべきでしょう。

さらに、適宜、ベンダーから、外注業者・下請業者の作業内容・作業状況についての報告をもらえるようにしておくべきです。

まとめ

以上、システム開発における下流工程の内容や注意点を検討しました。システム開発の成否の多くは上流工程で決まりますが、下流工程においても、変更管理やテストにおいて混乱したり、トラブルとなることもあります。システム開発プロジェクトを成功裡に終わらせるため、ユーザーは、下流工程においても、ベンダーに全てお任せという態度ではなく、ベンダーの作業をしっかりとチェックしていく必要があります。

次回は、こうした上流工程、下流工程の管理の実際を、現実にあった事例を基礎にして検討し、成功の秘訣を確認したいと思います。

第5回

成功事例に見る「成功のコツ」

牧野総合法律事務所弁護士法人（JIIMA顧問弁護士）　牧野 二郎
牧野総合法律事務所弁護士法人（弁護士）　森　悟史

これまで失敗事例（第2回）、基本的な原則（上流工程管理 第3回、下流工程管理　第4回）と見てきましたが、皆さんの中には、原則論は分かっているが、それを実践できないから問題なんだ、綺麗ごとでは現実はすまないんだ、そんな原則論で開発はできない、などの意見が聞こえてくるようです。

本当にそうでしょうか。原則どおり進めていたのでは開発はできない、というのは本当なのでしょうか。今回は、その疑問に回答すべく、実例をもって検討してみました。

大塚商会の開発事例

今回検討したのは、株式会社大塚商会が実施した開発事例で、ある建設資材卸業（以下A社とします）の注文を受けてA社の基盤となる業務システムを開発した事例です。この事例は、まさに原則どおりに丁寧な開発を実施し、無事開発が実施された事例です。特に原則を守るために、独特な工夫「フィッティングコンサル」という手法を実践して、成功に導いている点が見事でした。

大塚商会といえば、コピー機やIT機器の販売、オフィス用品の通販などで有名で、テレビCMのファンも多く、多くの中小企業が同社のサービスを受けています。同社は、IT関連企業、物販企業としてのイメージが強いですが、実は、システム開発にも力を入れており、自社グループ製品として基幹業務システム「SMILE」シリーズを開発、提供していることが知られています。「SMILE」はすでに30年以上も多くの企業で使用され、バージョンアップ（現在は「SMILE　V」）しています。幅広い分野で利用できる基幹パッケージソフトとして成長してきており、クラウド化、仮想化などの最新の技術を取り入れたものとなっているとのことです。

今回ご紹介する事例も、同社の基幹業務システムである「SMILE」を導入するにとどまらず、結果としては「SMILE」をカスタマイズし、更には関連プログラムの開発も含む大掛かりなものとなりました。

A社は創業70年を超える企業で、すでにパッケージソフトを利用して業務を遂行して来ました。そのシステムの老朽化を機に、新たなシステムの導入を検討され、大塚商会に相談したのが始まりのようです。

相談を受けた大塚商会は、当初パッケージの導入を基礎にしましたが、その後の丁寧な作業の中で、大幅カスタマイズが必要であると判断し、本格的な開発になった事例でもあります。結果として、A社の現在の建設資材卸の業務処理フローを維持することを前提として、必要なカスタマイズ、及び追加プログラムの開発を行うというもので、現行システムの処理フローを原則としてそのまま開発の基礎にすえる方式、いわゆる『カスタマイズベース・アプローチ』を基本とするものとなり、契約し、実施されました。

この開発は大きなトラブルもなく、順調にすすみ、現在本格稼動していることから、文字通り、現在開発事業に求められている作業を丁寧に実施し、成功に導いた好事例といえるでしょう。

その成功の秘訣を、同社執行役員であり、技術本部アプリケーション担当の責任者でもある朝香信一さんにお聞きしました。

提案　大風呂敷は広げない

まず本件の場合、発注者であるA社の業務は8年前にシステム化して動いているものの、その後の8年間で業務内容が大きく増加し変化したにもかかわらず、システムの業務への対応が十分には成されていませんでした。更に決定的な欠陥として、そのシステムでは消費税改正対応ができないといわれたため、早急の改修が迫られていたという事情がありました。その旧システムにおいては、現場作業を進めるにあたり、随所でエクセルを使ったり、人力での対応が発生しており、正確性に欠け、訂正が多く発生するなどの課題が山積していたといいます。

① 提案書の作成とレビュー

大塚商会は、企業の要望を詳細に聞き取り、また、提出されたRFP（提案要求書）を詳細に検討したうえで、営業担当と専門のSEが協議して、規模、作業内容、スケジュール、納期を確認して原価計算、収益などを考慮した提案書原案を作成し、規模に応じた社内幹部による提案内容のレビューを受けるといいます。大塚商会では、こうした見積もり作業においても、雛形が用意されており、その雛形どおりに提案書を作成することが求められていると言います。

この見積もりに基づく提案書の作成と、作成された提案書について責任者のレビューを経て、初めて客先に提案するそうです。こうした作業により、現場の独走を避け、実現可能な提案書で戦うことができるわけです。

ただ、提案が通らず、無駄になることも多いため、この作業自体は手間とコストのかかる作業ではあるけれども、手抜きしてはならない大切な作業として進められているといいます。

② 提案書は条件付で

同社の提案書には2つの特徴がありました。

一つは、「再見積もり条件」という記載があり、提案内容はその後の精査の結果、範囲や工数の変化により再見積もりとなることがあるということが明記されていました。フィッティングコンサルの後、あるいは要件定義の後に再見積もりするという合意の下で進めるのであり、最初から無理な価格での合意を取るような危険な作業はしないといいます。

競合する企業の中には、どう見ても無理な価格での提案を行い、大きなリスクがあるのではないかと疑われるケースも見受けられるといいます。大塚商会では、かつて現場だけで仕事を取りにいき、現場の判断で無理で安価な提案を行ない、痛い思いをしたことがあり、現在では本部SEの意見を確保し、レビューをした上でなければ提案を行なえないシステムにしたといいます。

二つ目の特徴は、「フィッティングコンサル」を条件付けていることです。「フィッティングコンサル」という業務は、同社独特な言い方のようですが、いわゆるF&G（フィットアンドギャップ）分析、さらにはパッケージソフトの体験導入による業務での、現実的な適用可能性の検討、及び不足部分の開発の必要性の吟味など、現場の方とのディスカッションや、ヒアリングを実施する作業を包括しているそうです。

大塚商会では、SMILEが現在の業務に適用できるかどうか、現業システムに対してパッケージでは足りない部分はなにか、現業を変える必要があるかなどの精査を行うという作業をお願いしていることです。SMILEの適用のみで進むケースや、カスタマイズが比較的少ない案件の場合にはそこまではしないそうですが、カスタマイズが大きくなる案件、作業が比較的大きなケースでは「フィッティングコンサル」の提案をして、ユーザーにその必要性を理解してもらい、合意して進めるというルールになっているといいます。

A社案件も、比較的大型のものであったことから、フィッティングコンサルが行なわれた案件という事になります。

「フィッティングコンサル」が終了すると、同報告書が作成・提供され、説明会も開かれて、ユーザーの理解が得られるようにしており、さらに、その後予定される開発作業の概要の把握も進むように工夫されています。

現場が独走すると、仕事欲しさに実現できそうにない無理な内容の提案をしたり、詳細を見ないでいい加減な約束をしたりして、結局開発が頓挫することにもなります。この様なことが起きないように、冷静な見積もりと、更にコンサルティング及び要件定義の実施の後に再見積もりすることをしっかりと合意してからすすめるという念の入れようでした。

数社が競合して、競り合った後、大塚商会が選択された場合であっても、直ちに開発に取り掛かるというわけではなく、まず、「フィッティングコンサル」が開始され、詳細が精査されることになるわけです。

「フィッティングコンサル」の有効性

フィッティングコンサルというのは、同社の基幹業務システムSMILEを実際に使用してもらい、その使用を支援しながら業務の合理化の道を探り、同時に、いわゆるF&G（フィット・アンド・ギャップ）を進めてゆく作業となります。

大塚商会では、自社開発の基盤ソフトであるSMILEを開発の基盤にしていることから、実際にSMILEを使ってもらい、その使用感を前提に開発に取り掛かれるかなどを、慎重に検討することを基本にしているとのことでした。こうすることで、実際にその使用感、良さを実感してもらい、あるいは現業の処理で足りないと思われるところ、不満なども聞き取り、現場での利用を前提とした詳細なヒアリングが可能となるというのです。

A社の案件で行なわれたフィッティングコンサルの場合、A社の現業務のヒアリングに11週間かけ、さらにパッケージ運用、カスタマイズの議論を8週間（ヒアリングの後半と重複）実施し、その後システム化の方針などの検討に6週間、新業務フローの提案・確認に1週間かけ、その結果をふまえて、フィッティングコンサルの終了報告を行っていました。5・5ヶ月、22週間か

けて行なわれたのです。フィッティングコンサルの担当者としては当初はコンサル1名で担当しシステム規模感がある程度把握できた段階からSE 3名を追加アサインして、現場での作業に入ったといいます。

この間の打合せの会議は40回弱行なわれており、各階層の皆さんに理解いただくよう、工夫をしていたとのことでした。

フィッティングコンサルの中間では、中間報告も実施され、それまでに明らかになった問題点、課題、開発費用の再検討、スケジュールの見直しなどの概要が示され、A社の開発担当との意識あわせも実施していました。

こうした作業の後に作成されたフィッティングコンサルの報告書は①現状分析、現在業務のフローの詳細と課題に明確化、②提案した新しい業務フロー、その目的と課題克服の詳細説明、③各業務部分におけるF＆Gの結果（機能要件対応、運用変更点など）のまとめ、④スケジュール及び開発担当者、開発体制の明確化という内容になっていました。

まさに、SMILEというパッケージを利用してはいるものの、カスタマイズベース・アプローチなので、パッケージのどの部分をどの様にカスタマイズしなければならないか、その要否、現業を担当している方々がその変化を受け入れられるかどうかなどを慎重に検討し、報告と説明を行い、A社の合意をしっかりと取り付けているという内容でした。かなり重厚なコンサルではありますが、開発費用全体から見ると5～6％程度に留まるのが多いそうです。

この初期段階（最上流過程）で行なわれるF&G分析について、その必要性と機能について、以下のように言われています。

IPA（独立行政法人情報処理推進機構）から出されている「システム再構築を成功に導くユーザーガイド」（第2版）では、わが国の開発事業者の基本的に守るべき手順として、以下のように規定しています。同書では、

> 「Fit&Gap分析は、企画・計画工程と要件定義工程のそれぞれで実施する。企画・設計工程では、パッケージ製品を決定し、プロジェクト計画案と概算見積もり算出のための分析を行う。」（同書29頁）

としており、最上流過程において、2回に渡り、実施すべきものとしています。最初に行なわれるF&Gすなわち企画・計画工程に行なわれるF&Gによって、初めて「概算見積もり」ができることになるという基本原則が示されています。企画・計画段階、すなわち要件定義工程の前の段階において、F&G分析を行うことで、見積もりをするための基礎となる情報、作業内容、開発対象、作業期間、概算の工数などが見え始め、はじめて概算見積もりが可能となるということなのです。

この基本に立てば、初期段階で詳細を検討せずに、明確な根拠も無く、見積もりの基礎も無く、漠然とした感覚のみで見積もりを出すことの危険性、いい加減さは明確です。このようないい加減な見積もりであれば、ユーザーから見れば、見積もりの根拠が全くといっていいほど存在しない見積もりを信用できるのか、という問題にもなります。

大塚商会の行なうフィッティングコンサルは、適正な「概算見積もり」を行なうための必須の作業を愚直に実践しているものであって、これが成功のための第一の秘訣であることが分かります。

合意を得て、作業を進めること

当たり前のことですが、開発担当者としては、ユーザーである発注者との合意を一つ一つ確認して、開発を進める必要があります。ユーザーは開発については素人ですから、開発担当者がどの様な開発を考えているのか、それはユーザーにどの様な影響を与えるのかなどを明示して、初めてユーザー側も開発内容を理解して、開発担当者の進め方に同意できる前提が成立し、適切な合意が成り立つことになります。

また開発者は、ユーザーの現場の作業について基本的に素人ですから、現場作業者から、現場の作業手順や作業方法などの話をよく聞いて、現場で使える、使いやすいシステムになるように工夫しなければなりません。こうした両者の作業が合致して、合意が形成されて、開発が進むわけです。

大塚商会は、まずこのフィッティングコンサルの報告の後、説明会を実施して、開発用内容を明確にして、この工程で得た情報をもとに、開発の方向性を示して、案件毎に開発チームの全員の合意を確保するといいます。こうして、第一段階の合意（報告書の検収）を得て、次の段階に進みます。

次は要件定義作成支援契約となり、更に開発設計（外部設計）契約、そしてそれに続いて構築（ソフトウェア開発）契約、移行支援契約へと進む仕組みになっています。

各契約においては、対象とする作業内容についてその対象を詳細に記載しており、どの様な作業を実施するのか、ユーザーの果たす役割と開発担当が果たす役割の明確化など、必要な事項はすべて書面に残し、合意して進むようになっていました。

安易にパッケージに頼らない

パッケージを大幅に利用する場合やパッケージをそのまま基礎にしている場合には、要件定義は不要とする考え方もあるようですが、これは疑問です。パッケージをそのまま導入し、一切の変更もなく、単なるパッケージ売買（利用許諾契約）を行なうのであれば、要件定義は不要と考えてもいいでしょう。

しかし、一部分であってもカスタマイズが入り、パッケージに無い機能を追加するために開発が必要である場合には、何を開発するのかを明確にし、他の機能との連携、パッケージとの連携を確保するなどの必要から、要件定義及び構築のためのF＆Gが必要となるわけです。そして更にF＆G分析で明確になった開発内容に沿って、要件定義を実施し、確保する機能を明確にしてゆくことが求められるわけです。

この点、重要なポイントがあると朝香さんが指摘します。

「多くの場合、社長をはじめ、経営陣の意見を聞くと、業務を見直す、改善する、新しいシステムにする、パッケージにあわせた業務にする、と意気込みをもたれるのですが、現場でお聞きすると、現場担当者は現業務の処理、フローを変える気はない、現在の業務を守って欲しい、というわけです。ユーザー企業の内部で、大きな対立が存在することが多いのです。開発担当者は、この点、対立している部分を理解した上で、丁寧に要件定義を行ないながら、両者の対立を解消して、円滑な業務が進むようにしなければなりません。この調整をしないと開発は成功しません。」

朝香さんのこの指摘は、本連載第2回目（2019年6月号）のスルガ銀行事件のスルガ銀行の経営陣の動きと現場の動きとそっくりのようです。日本IBMは上層部の話を鵜呑みにして進めると、現場が混乱して作業が進まないといった障壁にぶつかってしまった、というわけです。

このユーザー内部の対立現象をしっかりと把握して、全員の合意を得て、しっかりとした要件定義を行ない、それに基づいて進めるということが成功のポイントであるという指摘は説得力があります。

こうして、各段階で合意を確保したら、かならず各段階で検収が行われ、ユーザーからの検収書を確保しなければならず、大塚商会では、それらの検収書がない限り前に進むことができない仕組みになっているといいます。従って、ユーザーの合意を確保しないで、開発側が勝手に作業を進めたり、合意に基づかない勝手な想像で進むような危険は発生しないように制度化されているのです。こうして確保された合意、検収書の存在は、すべて管理システムに登録され、点検監査されています。

コミュニケーション、開発の管理

こうして行なわれた前述のヒアリングやF＆G分析の洗い出し、その結果説明においても、多くの時間を割いていることが伺えますが、すべての場面で、開発担当者とユーザーとのコミュニケーションの確保がきわめて重要なポイントになるといいます。

開発の最初の時点で、開発担当者とユーザーの開発担当者の割り当て、氏名による確定作業は確実に実施されます。それぞれ事前に自身の専門分野を前提に、開発責任者、各部門の担当者などを指定していますので、それぞれの担当者相互の定例会議が設定されるとともに、開発に関わる全体の意見調整のための合同会議なども設定されるといいます。経営層、管理層、更に専門分野ごとの担当者会議などが設定されるわけです。

更に大塚商会では、こうした会議の内容をすべて記録し、問題点を把握して、自社の基幹システムに組み込まれているPMS（プロジェクトマネジメントシステム）に登録することになっています。そして、毎週すべての開発メンバーに対して情報提供し、共有し、さらに毎月1回半日かけて進捗状況や課題などについての議論を行い、開発陣としての意見調整を行なうとともに、ユーザーとのすり合わせの状況も把握し、内部においても、顧客との間でも、徹底したコミュニケーションを確保することを制度化し、常時点検しているということでした。

こうしてプロジェクトをシステムに乗せて、管理することでコミュニケーションを活性化し、同時に、案件を個人に抱え込ませず、プロジェクトのリスクを全員が共有するという作業も可能となるということでした。

これに関連して、要件定義、その後の設計、開発作業の中でも、ユーザーからの要求の変更、要求の拡大などは、やはり頻繁に起きてくるといいます。しかし、これについても、常にユーザーの意見と開発側の意見の調整を行い、課題として開発担当者全員が共有する仕組みがあるので、ユーザーの変更要求への対応は的確に行なえるといいます。そして、リスクも明確になり、共有されているので、ユーザーに対するリスク告知、すなわち予算拡大やスケジュール変更などについても適切に説明し、同意して進める体制が確保されているといいます。同意できない部分は同意できない理由を明確にして、記録もして、説

明をするということです。ここでも、曖昧な結果にならない様に、全員で開発管理を行なっているわけです。

品質管理は徹底して行なうこと

大塚商会には、システム開発を専門とする子会社がある他、外注先として、いくつかの外部開発事業者と契約関係を維持していますが、こうした外注先の管理も重要となります。自社の管理は当然ですが、外注先に丸投げしていたのでは危険が多すぎるため、さまざまな工夫を凝らしています。こうした外注先の管理は、開発するシステムの品質管理に直結するため、重要視して進めているということでした。

外部に開発行為、構築作業そのもの委託する場合、法律的には請負であって、本来は、任せきりにするのが原則で、外注先の作業に関して細かく管理することはできないのですが、大塚商会ではまず、外注先としてふさわしいかどうかを慎重に判断した上で、詳細な契約を締結し、かつ日常的には定例会議を設けて、進捗状況の確認や開発メンバーの確認などを行うなど、下請け管理は慎重に行なっているとのことでした。

請負の場合には、作業完了時に成果物が納品されますが、大塚商会の場合には、外注先からの納品を検査する専門部隊、テストチームが編成されており、常に徹底したテストを実施しているといいます。成果物となるプログラムについて、単体テストを実施し、成果物の品質をチェックし、バグ率を確認して、指導しているそうです。バグを修正させるだけでなく、バグが増加することがないように適宜指導を強化しているそうです。こうした徹底した外注管理は、システム開発の品質管理に直結するため、重要なポイントとなっています。

移行には計画と立会いが必須

各種のテストを経て、システムが完成してもそれで終わりではなく、それからデータ移行を行い、並行稼動して、問題なく稼動していることが確認されて、初めて完了となるわけです。朝香さんは、この点でも注意が必要だと言います。

現場は、現在の業務が中心であり、それが忙しいので、新しいシステムを導入しようとしても、進んでは対応してくれないことが多いそうです。ユーザーの作業現場では、現在の仕事が優先されるというわけです。そこで同社では、最初にユーザーと一緒に移行計画を立て、誰が立ち会うのか、どの様な役割でどの様な作業を行うかを明確にして、さらにユーザーの受け入れの検収作業は、誰が、どのタイミングで、どの様にして行うか、それらについても明確にしたうえで移行作業をするそうです。具体的には同社の技術本部が陣頭指揮を執り、同社SEが中心となり、営業・開発・構築担当者など全員が、ユーザーの移行作業の対応をするように、指導しているといいます。

現場ではすでに長年にわたる慣行や古いルーチンがあり、新しい環境に慣れるまで時間がかかることもあるでしょう。新しい環境に慣れるという視点、意気込みができなければ、新しいシステムといえでも、なかなか十分な力を発揮できないでしょう。新システムの導入に当たっては、この移行の時期から、しっかりとした計画と対応で、現場を支えてゆく必要があるということでしょう。

成功の秘訣

以上、大塚商会がA社のシステムを開発して導入した案件を参考にして、現実にどの様な作業が行なわれてきたのかを見ることができました。そして、A社案件の資料は整理されており、また、管理システムの中ですぐに検索できるようになっていました。

こうした管理があることで、今後の保守、運用、更なる改修、改善に当たっても、十分な対応ができると環境が整えられているといえるでしょう。こうした記録があることは、ユーザーにはなかなか理解できないところでしょうが、大変重要なポイントであるように思いました。

システム開発の成功の秘訣として、今回のケースから学ぶべきは、次の点になるでしょう。

最上流工程（企画・計画）工程においては、ユーザーの意向を尊重した実現可能なシステムを構築するため、ユーザーの意向を徹底して汲み取り、かつ現場での現業の動きや、作業手順などをつぶさに把握して、ユーザーの業務とその課題をしっかりと把握することでしょう。その際に「フィッティングコンサル」は大変有効な作業であることが分かりました。

加えて、パッケージ製品を利用する場合であっても、要件定義をしっかりと行い、ユーザーの求める機能が確保されていることを確認し、開発作業を進めることが重要だということです。

下流工程（詳細設計、プログラミング、更には移行まで）においては、その多くが内部作業となりますので、品質管理がきわめて重要となり、要件定義、確定した仕様どおりの高品質なプログラムが完成するように、徹底した管理が行なわれることが重要とのことでした。

こうした作業の確認、点検、そして大所高所からのプロジェクトのマネジメントは、経験豊かなマネージャーの綿密な作業が必要とされ、その分経費がかかることになりますが、丁寧な作業により、初めてユーザーの望むシステムが完成するということであり、決して省略することができないものであることがわかりました。

システム開発を発注するユーザーにあっては、ユーザー毎に業務の個性があり、長い歴史の中で培われた独特のノウハウなどもあり、開発担当者の勝手な想像や原則論では対応できない現実があるようです。言葉にならないルールがあったり、見えていない手順があるなど、難しい部分が山ほどあるのでしょう。それに対して、愚直に、ユーザーの要望を聞くこと、ヒアリングを徹底することで実情を把握すること、しかし、同時にシステムが解決できること、対応できること、可能なことを明確にして、曖昧な対応をしないことが重要でしょう。その上で、経費、スケジュールなどについては、常に明確にして、十分な説明を行うことが求められます。

こうした作業の積み上げにより初めて、開発が成功するものであり、地道な対応により、実現するということが分かりました。

読者の皆様におかれても、この案件をぜひ参考にしてみてください。

本稿の作成に当たりお世話になりました朝香様始め大塚商会の皆様には心から感謝申し上げます。

次回は、多くの失敗事例や成功事例をふまえて、そのポイントを明確にしたうえで、開発を依頼する立場、すなわちユーザー目線での、失敗しないシステム開発を実現するための実践的なチェックリストを検討することにいたします。

御社の文書管理診断します！
文書管理達成度評価・調査ご協力のお願い

「皆さんの組織の文書管理のレベルはどのくらいですか？」

各組織では、内部統制、説明責任など、社会のさまざまな要請にもとづいて文書管理を実践しています。しかし、文書管理のレベルを測る仕組みがなく、これで十分なのか、不足している点は何かを知ることが難しいのが実情だと思います。

JIIMA文書管理委員会では、そんな疑問を解消し、各部門が正しく文書管理ができているかを診断するサービスを開始しました。貴社組織の現状を回答用シートに書き込み送付いただければ、文書管理委員会が診断しお返しします。

将来的にはご提供いただいた情報を元に、日本における組織の文書管理現状をまとめ、その中で各組織がどのレベルに位置づけられるかをわかるようにしたいと考えています。

自社の文書管理に関心がある組織の方々のご利用をお待ちしています。

メリット

- 自社の強みや弱みを明確に把握することができるとともに、取り組むべき方向性も明らかになり、文書管理の改善に結びつけられます。
- 他社のレベルと比較でき、自社の文書管理推進の動機付けになります。
- 一定の時間が経過した後に再評価することにより、自社の改善の度合いを確かめることができます。

詳細は右記URLを参照ください。 https://www.jiima.or.jp/basic/doc_mng/

第6回（最終回）

システム開発プロジェクトを成功させるためのチェックリスト

牧野総合法律事務所弁護士法人（JIIMA顧問弁護士）　牧野 二郎（まきの じろう）
牧野総合法律事務所弁護士法人（弁護士）　森 悟史（もり さとし）

前回まで、システム開発における成功例・失敗例、及び各工程におけるポイントを見てきました。

今回は、システム開発プロジェクトを成功に導くための重要ポイントをチェックリストにまとめました。システム開発プロジェクトはそれぞれ、開発対象、費用、スケジュール、納期、プロジェクトに割ける人員の数や知識・経験レベルが異なっており、画一的にまとめられるものではありませんが、一応の目安としてご利用いただければと思います。

開発を依頼する前に確認すべき事項（チェックリスト①）

ユーザーは、システム開発プロジェクトを開始する際、どの様なシステムが必要なのか、システム開発に割ける費用、人員はどれくらいで、本稼働のリミットはいつまでか（サービスインはいつか）などを明らかにする必要があります。これは、企画段階でのシステム化構想、システム化計画となります。

チェックリスト①は、システム開発プロジェクトを開始する際に、ユーザーが確認すべき事項の例です。ただし、この段階では、プロジェクトの内容が必ずしも確定しておらず、後の要件定義段階等で明確にされるものも含まれていますので、ここでは、ざっくりとした内容となっても構いません。

なお、開発を依頼する前に確認すべき事項は、ベンダーに提供するRFP（Request for Proposal・提案依頼書）のベースにもなります。ベンダーは、ユーザーから提供されたRFPをもとに、どのようなシステムを開発できるのかなどの提案をユーザーに行いますので、ユーザー側もしっかりとしたRFPを作成する必要があります。そのためにはシステム化構想、システム化計画をしっかりと詰めておく必要があります。

開発を依頼する前に確認すべき事項（チェックリスト①）

大分類	小分類	注意点
目的	導入の目的	システムで何をしたいのか、何を解決したいのか
開発対象等	開発する機能（開発範囲）	どのような機能が必要かなど （ただし、要件定義工程を経て正式に確定する）
	システム環境	オンプレミスか、クラウドかなど
	バージョンアップへの対応方法等	実施時期やその内容など
	データ連携	他の既存システムとの連携方法など
	バックアップ	バックアップ体制・方法
サービスレベル	稼働率（可用性）の目標	ただし、要件定義・外部設計工程を経て詳細が決まる
	許容される障害回復時間	ただし、要件定義・外部設計工程を経て詳細が決まる
	年間の許容障害件数	ただし、要件定義・外部設計工程を経て詳細が決まる
	許容される障害通知時間	ただし、要件定義・外部設計工程を経て詳細が決まる
セキュリティ	セキュリティ要件	ただし、要件定義・外部設計工程を経て詳細が決まる

開発方法	求める開発手法	ウォーターフォール型か、アジャイル型かなど
	検収の時期及び方法	
	テスト	テストの内容及び方法 誰がどのようなテストを行うのか（作業分担） 誰がテストケースを作成するのか ただし、要件定義・外部設計工程を経て詳細が決まる
開発体制	体制の概要	PM、チームリーダーなどの全体体制図 ただし、要件定義工程を経て詳細が決まる
	ベンダーとユーザーの連絡・報告方法	
	再委託の許否	再委託を許可する場合の条件も含む
	変更管理の方法	変更する場合の手続や費用負担はどうするのかも含む ただし、要件定義工程を経て詳細が決まる
パッケージ	パッケージ利用の有無	パッケージを利用する予定がある場合
	パッケージの内容・特徴	パッケージを利用する予定がある場合
	パッケージでカバーできる範囲（機能）	パッケージでカバーできない範囲（カスタマイズ部分）も含む
	パッケージの権利関係	契約の種類（利用契約かなど）
	パッケージにかかる費用	利用料（ランニングコスト）の上限・目安も含む
	パッケージの保守体制	保守費用の上限・目安も含む
移行・受入支援	移行の方法	ユーザーで行うのか、ベンダーに支援を求めるのかも含む
	受入支援	ベンダーに受入支援を求めるのか 受入支援を求める場合の支援内容など
スケジュール	おおよそのスケジュール感	各工程の終了時期を含む
	進捗管理の方法	
	スケジュールの調整方法	スケジュールが遅延した場合の対処方法など
費用	おおよその費用感・見積り	見積りの根拠を含む
契約関係	契約の方法	一括契約か、多段階の個別契約を締結するか
	契約の種類	請負契約か、準委任契約かなど
知財関係	特許権	成果物の特許権の帰属についてなど
	著作権	成果物の著作権の帰属についてなど
	FOSS（フリーオープンソースソフトウェア）	FOSS利用の有無 利用する場合の費用・責任関係など"
成果物	成果物の内容	納品してもらう成果物は何かなど
	設計書等のドキュメント	どのような書面を作成・納品してもらうのか 設計書の記載レベルなど
保守関係	メンテナンスの概要	他社へメンテナンスを依頼することがあるかを含む。
	メンテナンスの体制	
	メンテナンスの費用感	
拡張性	他社の利用	子会社やグループ会社において当該システムを利用するかなど
	他システムとの連携	他のシステムと連携させることがあるかなど
	機能追加・改変の要否	

ベンダーの選定基準（チェックリスト②）

いざ、ベンダーにシステム開発を依頼するとなった場合でも、ユーザーは、どのベンダーに依頼すればよいのか迷うのが通常でしょう。ただ単に、費用が安いといった理由でベンダーを選定すると、多くの場合、失敗します。システム開発を成功に導くためにも、あらかじめ、どのようなベンダーを選ぶのか、ベンダーの選定基準を策定して、この基準に従いベンダーを選定することが必要です。

チェックリスト②は、ベンダーの選定基準を策定する際に考慮すべきポイントをまとめています。

なお、ベンダーが選定基準を満たしているかどうか、ユー

ベンダーの選定基準（チェックリスト②）

大分類	小分類	注意点
会社の概要	資本金	参考程度
	設立年度	設立間もない会社の場合は注意
	売上	参考程度
	従業員数	マンパワーは足りているか
	業務内容	開発してもらいたい分野に精通しているか
	親会社・関連会社	参考程度
	提携先	参考程度
	資金繰り	倒産の可能性はないか
	ISMS等の認証取得の有無	参考程度
	これまでのインシデントの有無	
	インシデント対応時の計画の有無及びその内容	
組織	プロジェクト体制	重要
	プロジェクトに割けるベンダー側の人員	重要
	実作業員の実力・能力	重要
	実作業員の業務経験	重要
	代替要員の確保の有無	重要
	下請会社従業員のアサインの有無	重要、品質に影響することがある
経験	実績数	
	実績例	自社と同規模・同業種の会社における導入実績があるか
技術力	設計思想・方針	
	業務適合性	ユーザーの業務に適合するかどうか
	使用開発言語	保守性に影響する
	システムの環境	保守性に影響する
	パッケージの名称及び内容	パッケージを利用する場合に必要
	バージョンアップへの対応	保守性に影響する
	移行・導入支援の有無及びその内容	
	保守対応の有無	保守について他の業者に依頼する必要があるか 保守費用の費用感なども含む
	システムやサーバーの稼動状況	機密性・完全性・可用性に問題は無いか
	セキュリティレベル	セキュリティポリシーはどのようになっているのかなど
その他	開発費用	安すぎても駄目
	開発工数の妥当性	少なすぎても駄目
	ベンダーが考えているスケジュール感	ユーザーが求めるサービスイン時期に間に合うか

ザーにおいて確認するためには、ベンダーから情報を提供してもらう必要があります。そのような場合、ユーザーは、ベンダーに対しRFI（Request for Information・情報提供依頼書）を出し、ベンダーに情報提供を依頼することも必要でしょう。ユーザーがRFIを出しているにもかかわらず、ベンダーが的確な情報を提供しない場合は、そのようなベンダーを利用するのは止めるべきでしょう。

要件定義終結において確認すべき事項（チェックリスト③）

第3回でも検討しましたが、要件定義は、ユーザーがシステムに要求する機能を定義する作業のことで、ベンダーは、この要件定義をもとに外部設計書や詳細設計書を作成し、プログラミングを行います。もっとも、システムの素人であるユーザー自らが要件定義書を作成することは難しいので、ベンダーに要件定義書を作成してもらうことがあります。この場合、ユーザーは、要件定義書に自己の要求が適切に記載されているか、慎重に確認する必要があります。ベンダーはシステム開発の専門家であるから大丈夫だろうと安易に考えるのではなく、理解できるまでベンダーに質問し、場合によっては、他の第三者（システムの専門家）に要件定義書を見てもらうことも必要でしょう。

チェックリスト③は、ユーザーが要件定義書を承認してよいか判断するためのリストです。ここに挙げられている事項が要件定義書に記載されているかどうかを確認してください。

なお、チェックリスト③は、チェックリスト①の内容と重なる

要件定義終結において確認すべき事項（チェックリスト③）

大分類	中分類	小分類	内容	注意点
契約要件	パッケージ		パッケージ利用の有無	
			利用予定のパッケージの内容・特徴	
			パッケージでカバーできる範囲（機能）	パッケージでカバーできない範囲（カスタマイズ部分）も含む
			パッケージの権利関係	利用契約かどうかなど
			パッケージにかかる費用	利用料（ランニングコスト）も含む
			パッケージの保守体制	別途、保守費用がかかるかなども含む
	知的財産		特許権	成果物の特許権は誰のものになるのか
			著作権	成果物の著作権は誰のものになるのか
			FOSS（フリーオープンソースソフトウェア）	FOSS利用の有無 利用する場合の費用・責任関係など
	成果物		成果物の内容	納品予定の成果物は何かなど
			設計書等のドキュメント	どのような書面が作成され、納品されるのか 設計書の記載レベルなど
要件定義	全体方針		システム化の目的・目標	経営課題を解決できるシステムとなっているか
			開発の範囲	開発しない範囲も明らかにする
			開発手法	ウォーターフォール型か、アジャイル型かなど
			スケジュール	各工程の終了時期を含む
			進捗管理の方法	
			サービスインの時期	
			体制図	PMや実作業を行う者などの全体体制図
			ベンダーとユーザーとの連絡・報告方法	
			再委託の許否	再委託を許可する場合の条件も含む
			変更管理の方法	変更する場合の手続や費用負担についても含む
	機能要件	業務要件	対象となる業務フロー	
			入出力情報	どのような情報を取り込んで、どのような情報を吐き出すのか
			業務機能の一覧	
		システム要件	システム環境	オンプレミスか、クラウドかなど
			ハードフェア構成及びソフトウェア構成	
			関連する他のシステムとの関係	
			使用する言語	保守性に影響する
			アウトプット	画面、帳票等
		インターフェース	システム間のインターフェース	
			ヒューマンマシンインターフェース	
			接続端末のインターフェース	通信関係
	非機能要件	可用性	システムの稼働時間など	システムの停止予定も含む
			障害・災害時における稼働目標	許容される障害回復時間、年間の許容障害件数、許容される障害通知時間、復旧・回復方法等を含む
		性能・拡張性	システムのパフォーマンス（性能）	処理能力、レスポンスタイム等（高負荷時も含む）
			システム追加の容易さ（拡張性）	他社システム・ソフトウェアの利用は可能か
				後日、他のシステムと連携させることは可能か
				容易に機能追加・変更できるか
		運用・保守性	稼働レベル	監視手段・バックアップ体制を含む
			保守（メンテナンス）について	メンテナンスの概要
				メンテナンスの体制
				メンテナンスの費用感
		移行性	新システムへの移行期間・移行方法	誰がどのような作業を行うのかを含む
			移行対象資産の種類・移行量	
		セキュリティ	セキュリティ要件、対策・方法	重要
		システム環境・エコロジー	耐震・免震、重量・空間、温度・湿度、騒音等、システム環境に関する事項	
			CO2排出量や消費エネルギー等、エコロジーに関する事項	
その他	テスト		テスト内容	テストの内容及び方法
				誰がどのようなテストを行うのか（作業分担）
				誰がテストケースを作成するのか
				テスト終了基準（システム完成基準）
	受入支援		受入支援の有無及び内容	ベンダで受入支援を行うか 受入支援を行う場合の支援内容など

＊非機能要件の分類は、IPA（独立行政法人情報処理推進機構）の「非機能要求グレード2018」を参考にした。

部分も多いですが、要件定義終結段階においては、開発の対象や具体的な機能等の内容が固まっていることから、さらに緻密に確認する必要があります。

経営者がプロジェクトを中止するか判断するためのチェック表（チェックリスト④）

システム開発は、経営課題を解決するためになされるものです。そのため、システム開発を進めるにあたっては、プロジェクトにおけるマイルストーン（大事な節目）において、経営者による確認・判断が必要となります。経営者は、経営のプロですが、システム開発に精通しているとは限りません。そのため、システム開発プロジェクトについて、現場にまかせっきりにしてしまうことが多々あります。しかしながら、システム開発は経営課題を解決するためのものであり、経営戦略と整合する必要があります。経営戦略を明らかにし、システム開発が経営戦略に合致し

経営者がプロジェクトを中止するか判断するためのチェック表（チェックリスト④）

この表は、経営者がプロジェクト中止・見直しの決断を下す際に考慮すべき事項をスコアで表したものです。
スコアは当該事項の問題の大きさ・深刻さを示しており、1が一番軽微な場合で、5が一番深刻な場合となります。
網掛け部分は、当該事項が発生した場合に通常想定される問題の大きさ、深刻度合いを示してます。スコアが5や5に近いと判断される場合や、複数の事項が重なりスコアが大きくなるような場合には、直ちにプロジェクトを見直し、状況に応じて中止を決断する必要があります。

	発生した事項		スコア				
			1	2	3	4	5
1	スケジュール遅延						
	（大幅遅延の評価）						
	・期限・納期に間に合わない						
	・延期することは可能						
	（遅延の理由）						
	・発注内容の増加						
	・ベンダーとユーザーの意見の食い違い						
	・ベンダー側の事情・能力不足						
	・ユーザー側の事情						
2	開発費用の増加要求の評価						
	・発注内容の増加						
	・見積もりのミス						
	・詳細設計等で工数の増加を発見						
	・ベンダー側の事情・能力不足						
	・ユーザー側の事情						
	・増加の程度	1～2割増加					
		3～4割増加					
		5～7割増加					
		倍増					
3	機能不足						
	・必要な機能の欠如						
	・最重要機能の欠落						
	・重要機能の欠落						
	・複数の機能の欠落						
4	障害・バグ・フリーズなど						
	・障害・バグがある						
	・障害・バグの修正ができない						
	・フリーズ、ダウンする						
	・誤作動						
5	性能						
	・応答が遅い						
	・処理能力が低い						
	・負荷に耐えられない						
	・バックアップがとれない						
	・セキュリティに問題がある						
	・使用できない機器がある						

ているかどうかを検討した上で、システム開発プロジェクトにおける最終判断を行うのは、経営者の責任です。

また、システム開発プロジェクトには、さまざまなトラブルが発生します。そのトラブルを適切に対処するのは当然として、仮にトラブルが大きくなった場合、システム開発プロジェクトをどのように舵取りするのかを決定する必要があります。状況に応じて、プロジェクト途中で中止を決定する必要もあるでしょう。この最終的な判断権者も経営者です。

途中まで費用をかけてプロジェクトを進めてきたのだからと、なかなか中止を決断できないこともあるでしょう。しかしながら、これは、コンコルド効果（コンコルドの誤謬・サンクコスト効果などとも言います。）というもので、絶対に避けなければなりません。コンコルド効果とは、超音速旅客機コンコルドの開発につき、このまま開発を進めても採算がどれないことが想定されていたにもかかわらず、それまで多額の費用を投資してきたことから開発を中止することができなくなった。その結果、さらなる損失を膨らませ、破産してしまったという実例から、今まで投資したものが無駄になるという理由で、そのまま続けても損失が出るだけなのに止められないという心理現象のことです。システム開発プロジェクトでも、ここまで進めてきたのだからと言ってズルズルと引き伸ばしていたのでは、損失を拡大させるだけで、何も得るものがありません。経営者は、適宜適切に決断を下さなければならないのです。

チェックリスト④は、経営者がプロジェクト中止の決断を下す際に考慮すべき事項をスコアで表示しています。スコアが高いほど、プロジェクトの中止を検討する必要があります。

最後に、これまで6回にわたり、失敗しないシステム開発について検討してきました。紙面の都合上、説明が不十分となってしまった箇所や検討できなかった論点もありますので、何かご不明な点や質問等がございましたら、以下までご連絡いただきたく存じます。

牧野総合法律事務所弁護士法人
メールアドレス：info@makino-law.jp

本日紹介したチェックリスト①～④は下記URLよりダウンロード可能です。
https://www.makino-law.net/information/ダウンロードサービス/

copyright©2019　牧野総合法律事務所弁護士法人　All Rights Reserved

御社の文書管理診断します！
文書管理達成度評価・調査ご協力のお願い

「皆さんの組織の文書管理のレベルはどのくらいですか？」

各組織では、内部統制、説明責任など、社会のさまざまな要請にもとづいて文書管理を実践しています。しかし、文書管理のレベルを測る仕組みがなく、これで十分なのか、不足している点は何かを知ることが難しいのが実情だと思います。

JIIMA文書管理委員会では、そんな疑問を解消し、各部門が正しく文書管理ができているかを診断するサービスを開始しました。貴社組織の現状を回答用シートに書き込み送付いただければ、文書管理委員会が診断しお返しします。

将来的にはご提供いただいた情報を元に、日本における組織の文書管理現状をまとめ、その中で各組織がどのレベルに位置づけられるかをわかるようにしたいと考えています。

自社の文書管理に関心がある組織の方々のご利用をお待ちしています。

メリット

- 自社の強みや弱みを明確に把握することができるとともに、取り組むべき方向性も明らかになり、文書管理の改善に結びつけられます。
- 他社のレベルと比較でき、自社の文書管理推進の動機付けになります。
- 一定の時間が経過した後に再評価することにより、自社の改善の度合いを確かめることができます。

詳細は右記URLを参照ください。　https://www.jiima.or.jp/basic/doc_mng/