事例番号	事例タイトル	概要	事例における見切りの内容	本来の見切りの考え方(※)	捉えるべき兆候	対処例
1	**顧客側担当者が異動になり約束事が反故に！**	顧客側の担当課長レベルと費用および作業内容について内々に話をつけ仕事をしたが、顧客側の職制が変更になった。(前任者は退職)後任者は前任者からこのことについて全く引継ぎがなかったため今までの話がご破算になった	決められた社内ルールを無視して顧客との契約をせずに開発に着手	・費用と作業内容については最低限、文書で取り交わしておく。作業を実施するに至った経緯についても記録しておき、たとえ後付けでも作業の妥当性を顧客側の新担当者に説得できるようにしておく必要がある ・その非公式な約束事が反故になるケースを想定し、そのリカバリ方法が確保できていて、かつリカバリコストへの対策を考えておく	顧客側の人事異動情報	・契約する前であれば、機能削減し、予算判明以降に発生する赤字を減額する ・契約内容を確認し、プロジェクト単体が最小限の損害で契約を満たす方法を検討する ・システム化対象範囲を分割して複数年度の契約にするなど、戦略的に採算が取れるような交渉を進める
2	**本当のユーザーを見誤ってしまう**	情報システム部門からの要請によるシステム構築であり、情報システム部門と契約し、開発を進めていた。そのため、顧客内部のユーザー部門と情報システム部門に対立があったようだが、発注者である情報システム部門主導でのシステム仕様作成を進めていた。いざ実装してユーザー部門での利用が始まると、操作性や仕様に関しての問題が相次ぎ、仕様変更対応に追われ、予定通りの本番稼働が出来なくなった	ユーザー部門からの旧システムに対する問題提示や新システムへの要求等が上がっていたが、あまり議題に取り上げることなく情報システム部門主管で仕様決めを行った。本来はユーザー部門側からも仕様検討者がいるべきではあるが、顧客企業の情報システム部門ということで、特に仕様に関しては問題ないだろうと思っていた	・よい考え方とは言えないが、受注側のプロジェクト・マネージャーは、発注者側との検討の末に定義した仕様について仕様凍結であることを合意できていれば、その後の仕様変更に対して別途契約を結ぶなどの対処を取ることができる。そのために仕様凍結の議事録を残すなどの処置は、プロジェクト遂行上、最低限必要である ・情報システム部門だけで業務をきちんとサポートできるシステム仕様を決定できるかどうかは、その担当者の業務経験量や知識量に依るところが大きい。システム部門の担当者が業務部門から信頼が厚いことが望ましい ・一般的にエンドユーザーであるユーザー部門から「これでは業務が回らない」と新システムに対する文句が出ると情報システム部門は対応せざるを得ないことが多い。したがって上流工程においては、仕様の決定に際し、情報システム部門だけではなくユーザー部門側の承認ももらうようにする必要がある	・システム機能の話ばかりで業務レベルの検討を実施しようとしない ・先方の体制にユーザー部門が入っていない ・情報システム部門がユーザー部門との打ち合わせの場を持ちたがらない ・ユーザー部門への質問を情報システム部門で制限をかけようとする	・はっきりと仕様変更だと認めて頂いた上で、仕様変更の手続きに従い、計画の修正／変更を行う。まだ上流工程であれば手戻り量は少ないことが多いが、仕様を見直すとスコープが想定以上に拡がることもあるので、場当たり的な対処はなるべく避ける ・仕様凍結が公式に認められていない場合、更に要件定義／基本設計から請けており、発注者側が「仕様検討時の問題である」と一歩も譲らない場合など、要望を随時聞いて対応するのではなく、ユーザー部門に対する要求仕様の洗い出しから整理しなおす。場当たり的な対応では、仕様矛盾が埋め込まれ、その後の工程で問題が顕在化することが多い
3	**新サービスと非合理的な納期**	金融機関で創立〇〇年の記念日に新サービスを開始することが至上課題だった。新サービスについてはマスコミへの発表もされていたため納期順守が絶対であった。ところが、新サービスということで実現方式も二転三転し、手戻りが発生。カットオーバー前に突貫作業でとりあえず動くレベルの物を納入。しかし、稼働後は突貫作業の影響で品質の問題が多発した	目新しい新サービスということで、実現方式がなかなか決まらなかった。そのため、非機能要件やセキュリティポリシーについてもなかなか決まらず、方式設計が曖昧なままであった。納期まで時間がないこともあって、開発に進めざるところから着手していかざるを得なかった	・プロジェクト進行上何か問題が発生した場合は、QCDのいずれかを犠牲にするしかない。つまりQ(品質)、C(コスト)、D(納期)のどれを優先させるかを明確に意識するべきである。この事例ではD(納期)を優先課題としているため、Dを犠牲にはできない。したがって、Q(品質)かC(コスト)を犠牲にする必要がある ・上流工程の段階で不調が現れてきた場合は、最悪のシナリオを考えて、実現しなければならない最低限の機能と品質について早めに合意を取ること ・中・下流工程であれば、予算を追加し、要員を増やしたりすることは可能であるが、上流工程では要員の増員は功を奏しないことが多い。したがって、Q(品質)をどこまで妥協できるかを顧客側と合意が取っていれば、プロジェクト側も対策を打ちやすい	・マスコミ発表など対外的なセレモニーを優先させるキーパーソン ・顧客任せの仕様決め	・顧客側でなかなか仕様が決まらない場合は、納期を基点にして、バックフォワードで線表を引き、これ以上の遅れはプロジェクトの破綻を招きかねないという危機感をもってもらうようにする。そのためには類似する他プロジェクトを参考にするなどして、なるべく詳細なWBSを準備する ・顧客側の上層部に対しては、マスコミ発表のキャンセルについて手続を検討しておいてもらう(これにより現実的な危機感を持ってもらうことができる) ・方式がある程度限定可能な場合はそれを満たすマルチスキルな人材を確保し、次工程に向けた要員確保と準備を進める
4	**仕切る気がないとりまとめ役**	大きな開発物件で、複数ベンダーが一緒に開発を受注した。開発範囲の中でどこまでをどのベンダーが構築するかという分担が中々決まらずに時間だけが経過。責任分担が不明確な機能の重複や、機能自体が無い物が発生してしまった	会社間での調整が不十分であることがわかっていたが、開発が進むうちに開発分担が決まっていくだろうと思い、とりあえずプロジェクトは開始された	・開発の境界線が曖昧なプロジェクトは破綻する危険が高い。特に複数ベンダー間での調整の場合、なるべくうまみのあるところで境界を引きたがるため、とりまとめ役(この事例では発注者)の仕切りの技量が問われる ・とりまとめ役がどの程度プロジェクトを仕切る気があるのか、どれだけのオーナーシップをもってプロジェクトに参画しているかを見定めて、十分な仕切り能力があるのであれば、プロジェクトの進行とともに開発範囲も明確になる可能性があると見てよい ・発注者側が仕切り役であるという役割分担を認識していないようであれば、プロジェクト体制や役割分担について顧客と合意を得た上で、想定する境界線を自ら宣言してプロジェクトを進める必要がある	・主張しない発注者(ベンダー任せ) ・会議体を設定するだけの発注者 ・検討項目や課題項目だけがリストアップされて解決担当社(者)が明確にならない課題管理	・プロジェクト全体の俯瞰図を作成し、境界と見逃しの有無を確認する ・自分の作業範囲の確定と、自分に関係する作業範囲を誰が実施するのかを明確にする ・請け負ったシステム開発の範囲について想定範囲を明示し、その開発を進めるために必要な情報の提供を積極的に打診する。
5	**発注者側から要件が小出しにされ、マネジメントできなくなる**	発注者側にて要件定義を実施しており、その要件定義の結果を受けて、受注者側で開発することになっていた。しかし、サービスインのスケジュールまでの時間に余裕がなく、しかも要件について小出しにされ、提示された要件の変更も多発。基本設計の手戻りも多発することになり、結局赤字プロジェクトになってしまった	要件定義が未完了ではあったが、要件定義が固まったと顧客から言われたところから、基本設計に着手せざるを得なかった	・工程を後ろから引いて、要件定義をフィックスしなければならない日が明確になった上で、その日までに顧客から要件について提示があることを作業の前提条件とすることを合意しておく ・重要な機能とそうでない機能があるはずであり、その優先度を意識して要件定義が進められていることが望ましい	・発注者側の要件定義の進捗状況 ・キーパーソンが要件定義状況の全体を語れるかどうか(決定済と未検討項目について説明ができるか)	・まずは発注者側で洗い出そうとしている要件の検討範囲や検討項目を教えてもらい、受注者側ではある程度の作業ボリュームをおさえておく ・要件未確定によるリスクの洗い出しを実施し、発注者側に対し合意してもらう ・要件定義をフィックスする期限を明確にし、顧客に約束して頂く。この際、すべての要件定義を何日までに、という形では具体性がないので、先に洗い出した要件の検討範囲・検討項目に照らし合わせて、各項目毎に優先度の高い順に期限日を設けてマネジメントを行う
6	**体制の事前確保が仇に！**	契約に向けて顧客側と折衝を続けていたが、金額面がなかなか折り合わなかった。顧客側から要求されている納期が非常に短く、サービスインスケジュールを確保するためにベンダー側で要員を確保した。要員がそろったため作業を進めていたが、いざ契約の段階で開発対象機能を削減されてしまい、それまでに進めていた基本設計作業は契約範囲外ということでなかったことにされ、契約未締結期間の要員コストはベンダー側の持ち出しとなり、赤字プロジェクトになってしまった	プロジェクトを進める上で必要な要員をすぐに準備できるわけではない。契約を締結してから体制作りや要員調達をしていては、短納期プロジェクトは難しい。そのため、契約は未締結であったが、金額交渉に問題はないと判断し、基本設計に着手してしまった	・契約未締結であるにも関わらず作業を実施することは認められていない ・契約未締結期間の持ち出し分についてのコストを事前に見積もっておき、その分のコスト見積もりと実施内容について、契約時に上乗せさせてもらうことを合意してもらっておく ・契約締結の遅れは営業の責任と割り切れるのであれば、営業部門と交渉し、持ち出し分の費用について負担してもらう確約を取る。そのためにも、プロジェクトの悪化要因が契約プロセスにあることを営業や顧客に理解してもらっておくこと ・金額面で折り合いがつかない状態であることから、費用削減交渉がされており、当然機能削減や最悪の場合プロジェクト中止の場合もあることを想定しておかなければならない	契約期間の遅れとその原因	・契約の締結を急ぐ ・作業着手依頼を頂く ・依頼単位で作業承認を頂く ・作業時間の記録を残す ・上記によって、仕事を依頼しているという認識を共有する ・金額が折り合わないにも関わらずサービスインスケジュールは守らねばならない理由があるはず。見積り前提が崩れるなら、それを補填可能な要求を出すように折衝を進める ・契約締結・金額アップは営業をフォローする ・原価低減活動(オフショア、共通部品化など)を検討しておく
7	**営業がプロジェクト・マネージャの承認のないまま概算見積もりで受注契約を結んでしまった**	顧客側のシステム投資予算が決まっているようであり、ベンダー側の営業部門はその予算額をある程度把握しているようだった。その予算に収まるよう、概算で予算を見積もって顧客に提示し、契約した。しかし、最終的には見積もりを大きく上回る開発工数を要し、大幅赤字に追い込まれた	営業が詳細見積り未完了のまま請負金額を確定させてしまっており、プロジェクト・マネージャーによる見積もりを実施しないまま開発を着手してしまった	・プロジェクト・マネージャがプロジェクトを請け負うときに、自らが見積もった金額を営業に提示し、営業側見積もり予算とプロジェクト側の見積もり予算との乖離を明示する。 ・その時点ですでに赤字であったとしても、それをコストのベースラインとし、それ以上の赤字を増やさないという視点でプロジェクト・マネジメントを請け負う。営業側見積もりの甘さによる赤字は、プロジェクト実施側の責任ではない。このことを営業と合意できているのであれば、プロジェクトを進めてもよい	・契約確定時の見積もり根拠 ・戦略受注(たとえ赤字になろうとも受注することに重きを置くという受注)かどうか	・まだ上流工程で問題が顕在化していないのであれば、プロジェクト・マネージャが見積もりをしなおして、早めに契約時の見積金額との差異を理由と共に明らかにする ・たとえ営業を詰めても「それを何とかするのがプロジェクト・マネージャだろ」と言われることが多い。可能なコスト低減策を説明し、それでもこれだけの低減が精一杯であるということを説明すること ・可能な限り生産性を挙げつつ、合理的な理由を掲げ

事例番号	事例タイトル	概要	事例における見切りの内容	本来の見切りの考え方(※)	捉えるべき兆候	対処例
8	基本設計の承認なしで詳細設計に着手	顧客がビルを移転するとともに古いシステムを廃棄することなっていたため、新ビルでの新システム稼働が絶対条件であり、サービスインスケジュールになんとしてでも間に合わせる必要があった。要件定義工程が大幅に遅れたため基本設計を顧客に提示後、承認を得ないまま、詳細設計まで進行。途中、顧客から基本設計の変更を要請され、詳細設計をやり直すという二度手間になり、遅れたスケジュールを取り戻すため、要員を追加することとなってしまった	要件定義に時間がかかったが、その分、システムの実装の姿が見えていたつもりだった。顧客側もだいぶ焦っており、基本設計書を顧客に提出したが、もともと承認されるまでに時間のかかる顧客だったので、承認未完了のまま詳細設計に着手した	・詳細設計書に顧客の合意がないまま着手できるのは、その着手範囲の仕様変更が発生してもリカバリができるだけの範囲である。たとえば基幹系の重要な部分や他システムとの連携が多い部分は、仕様がひっくり返されれば影響範囲が広いため、たとえ工程が遅れても合意を得た上で着手する必要がある ・仕様の固まり具合を見定めて、システム単位、機能単位、データ単位などのいくつかの視点で、仕様変更に柔軟なシステム設計をする工夫をする必要がある。その見極めを行い、仕様変更をして欲しくないところを説明できるようにしておくこと	基本設計書に顧客側の承認があるか	まずは顧客から要望されている基本設計レベルの仕様変更が、すでに進行している詳細設計に対してどの程度の影響なのかを調べる必要がある。基本設計書の合意を得ずに進めている以上、受注者側は文句を言える立場ではない
9	評価が終わっていない基本設計で協力会社に着手依頼	基本設計のフェーズが若干遅れてしまい、まだ品質評価（フェーズ終了判定）が終わっていなかったが、詳細設計を協力会社に着手してもらった。その後、下流工程で不適合箇所が発見され、原因調査の結果、基本設計不良であるとこがわかり、大幅な手戻り作業が発生してしまった	詳細設計以降を委託する予定だった委託先とはすでに契約済みだった。基本設計の品質評価(完了判定)が未完了ではあったが、その委託先の着手日が到来してしまったこともあって、その基本設計を提示し、詳細設計に着手してもらった	・先出しする仕様の確定度がどの程度かをよく吟味すること ・先出しする仕様に変更があった場合に、どの程度の手戻りが発生するかを把握できており、たとえその手戻りが発生してもリカバリできると考えているのであれば先出し着手でもよい	基本設計の出来具合を、直接執筆者にヒアリングし、問題意識を感じているかどうか	・要件の確定度が高く、基本設計の品質の高い機能から詳細設計に着手。その際、その部分の仕様変更が発生した場合のコスト増については自社がかぶるものという認識と覚悟が必要 ・協力会社との契約の見直しが可能であれば見直しを行い、着手日の延期を調整する。それにともなう費用と、仕様変更による費用との比較を行って、よりコストの低い方法を選択するという考え方もあるが、仕様変更は別のリスクを生むため、なるべくならお金をはらってでも、着手を待ってもらうべきである
10	性急な立ち上げでの体制確保のため、要員調達でコスト高に	スケジュールを守るために開始時期をずらせなかった。通常は設計工程は社員比率を高くして実施するが、社員を集めるためには社内調整に時間がかかるため、協力会社の要員で補充することにした。ただ、急を要する手配だったことと、ハイスキルの要員補充をお願いする必要があったため、コストをあまり下げることができなかった	プロジェクトの立上げに手間取り、設計フェーズでの要員の確保が不十分のまま、設計を開始した	上流工程は人に依存する部分があり、上流工程の途中で人を増やしても効果が出ないことが多い。したがって、社員か社外（協力会社など）の要員かに関わらず、上流工程の立ち上げ時点からべったり参画できるかどうかが重要	欲しい人材（社内）の業務負荷	さらに次の工程の要員確保を早めに始め、超過分を補うべく有利な条件で契約を行いマイナス分を吸収
11	業務経験・業務知識の乏しい要員で要件定義工程を実施	要件定義に必要な要員の人数は確保できたので、プロジェクトをスタートした。担当者に業務知識がなく、要件定義は日を追うごとに遅れていき、成果物の品質もきわめて低いものとなってしまった	業務経験・業務知識があるかどうかを特に確認しないまま、確保した要員ばかりで、要件定義作業に着手した	・業務経験・業務知識のない要員だけでの要件定義では、十分な要件定義の品質を確保することは相当難しい。要員のそれらのスキルを確認していないことがそもそも問題 ・たとえ要員の業務経験・業務知識の不足がわかっていたとしても、顧客側の担当者が業務経験・知識について深い理解を持っており、ベンダ側の業務知識不足を理解してもらっている上で仕様検討の打ち合わせをしてもらえる状況であり、かつ、その担当者によるレビューを頻繁に実施してもらえるのであれば、軟着陸させることができる	・体制作りの段階での要員のスキル確認 ・要件定義工程の手戻りの量 ・顧客側からの苦情	・なるべく早期にエキスパートに参画してもらうなどの体制面での強化を急ぐ ・要件定義のレビュー頻度を高めるなどのレビュー体制強化をする。その際のレビュアーは、顧客側の業務担当者とベンダー側社内のエキスパート ・要件定義書の標準や作成手引きなどを作成して品質を一定におさめる
12	コーティングの標準化を実施する時間を惜しんだためにかえって非効率に	段取り不足で、標準化を実施する時間的余裕がなかった。単体テストや結合テストで十分な応答速度を得られない処理が多数発覚したうえに、プログラム修正にも必要以上に時間を要することとなった	コーティング規約を作成しないで、プログラムコーディングを実施した	・開発担当者のスキルとモラルが高い場合は、コーディング規約がたとえ定められていないとしても、高品質でわかりやすいコーディングをすることが多い。その場合は、必ずしもコーディング規約が必要というわけではない ・レビューを徹底して品質を確保することができれば、規約はそれほど重要ではない。ただし、コーディング規約がないことによるコードのバラツキが、レビュー効率を下げることもあるので、開発担当者のスキルのバラツキがある場合には、やはり遅れてでも作成するべきである	開発担当者のスキルの低さ(コーディング規約のなさが生産性の低下につながる)	・コーディング規約は言語毎にインターネット上で入手することができる（組込ソフトウェア開発向けのC言語の規約についてはSECより提供されている）。自社でコーディング規約を策定する時間がない場合は、これらのコーディング規約を利用する ・プログラミングの工程にさでに入ってしまっている場合は、各自が最初に作成したプログラムに対し、全員でピアレビューを実施する。2つめに作ったプログラムも全員でピアレビューする。以降はプログラムの品質が一定になる ・静的チェッカーを利用する
13	法務部同士で合意に至らず契約が破談	発注側と受注側の、それぞれの法務部間で契約書の内容に関する調整が滞っていたが、設計に着手した。ところが、契約書の内容について合意が得られず、破談となってしまい、それまでの設計工数が丸々無駄になった	本来、未契約で作業を実施することはよくないことだとわかってはいたが、ただ単に事務処理が滞っているだけだと思い、設計に着手した	・契約前の作業については、その契約が破談になったときにそれまでの作業コストが自社で持ち出しになることについて認識しておくべきである ・契約前にも関わらず作業を実施しなければならない場合は、顧客からの作業着手依頼書等の証跡を残した上で、作業を開始すること	契約締結の遅れとその原因	契約締結までは絶対に作業に着手してはいけない
14	初物プログラム言語での開発で十分な性能が出ない	ユーザ側から、まだ市場に出て日が浅いプログラム言語を使っての開発を指定された。その顧客はそのベンダー側にとっては重要な顧客であったため、どうしても受注したかった。そのプログラム言語での実装を行ったが、十分な性能要件を満たすことができなかった	どうしても受注するという営業判断もあったため、自分たちで経験していない新しいプログラム言語についての見極めができていないまま、プロジェクトをスタートさせた	新しいプログラム言語での開発の場合、いざとなったときに外部から専門の要員を連れてくることも困難である。最悪のシナリオを考えて、慣れた言語での開発について、決定権のある顧客側キーパーソンとネゴシエーションをしておき、従来法での開発の準備をしておければよい	いざとなったときに頼れる専門家が外部にいるかどうか	設計中に別途チームを作って、新言語での実機能確認のためのプロトタイプを作成して確認する。同時に、この結果に問題がある場合の対処方法をユーザと合意する
15	現行システムと新システムの並行開発	新法制度への対応のため、現行システムを、そのシステムを保守しているベンダーがリプレースするというプロジェクト。基本は現行システム機能を踏襲することになっており、新しい法制度の要件を加味したシステムにする必要があった。 ところが現行システムも機能追加・変更が多く、現行システムへの機能追加・変更が、現行踏襲の新システムにも影響し、開発作業が重複するという事態になった	新システムの開発で現行システム踏襲分の仕様を凍結し、新システムの結合テスト後に、それまでの間に現行システムに発生した機能追加分を一括開発することにした。しかしそのため、総合試験環境と開発環境の構築が重複することになり、環境の構築、管理する要員の負荷が増え、環境ミスによる進捗遅れが生じてしまった	結合テスト後に、新システムに現行システムで発生した機能追加分を一括開発する際に、現行システムの保守メンバーを新システム構築のプロジェクト側にアサインできることを確認しておくこと	現行システムの障害発生状況	機能追加・変更に合わせて現行システム要員を更改システムに参加させる
16	運用テストに入ってから現場固有の業務要件が多数発覚	給与システムの更改プロジェクト。運用テストに入った段階で、実は現場でローカル仕様があることが判明。ローカル仕様の調査には、かなりの期間と工数が必要であったが、システム稼動後、ローカル処理を手作業でカバーすることにした。そのため、1年間にわたって、顧客側の運用コストが大幅にかかってしまった	現場でローカル仕様があることが運用テストで判明したが、他のローカル仕様の有無を調査せずに、システム更改を実施	ローカル仕様が膨大になることを想定して、新システムの移行後に、現システムに戻せるようにしておく	・設計工程でローカル仕様を含めた現場の業務調査を実施していない ・プロジェクト設計メンバーに顧客側業務部門が関与していない	ローカル仕様がどれだけ存在するのかを把握し、もしその量が多い場合は、段階移行かリリースの延期をするなどの検討を顧客と実施する

事例番号	事例タイトル	概要	事例における見切りの内容	本来の見切りの考え方(※)	捉えるべき兆候	対処例
17	標準仕様に先駆けてシステム構築したが、標準化への対応で品質問題発生	ある先進的なシステムを開発していたが、他社ベンダーで同様なシステムを開発中との情報があり、それより先に提供し稼動させる必要があった。そのシステムに関連するJIS化が予定されている国際標準もあったが、独自仕様で開発し、納期に間に合わせた。ただし、その標準への完全準拠のために改造を行ったが、品質に問題が多く、顧客からのクレームがあった	関連する国際標準のJIS化が進められていたが、その策定スケジュールと顧客側の納期が合わないため、独自の仕様で開発を進めた。あとでJISが策定されてから、改造すれば対応できると考えた	・競合社よりも先に稼働実績を作ることに重きを置くか、他社に遅れつつもJISに徹底対応した安定システムを稼働させるか、どちらの方針なのかを明確にする ・JISと独自仕様がどのくらいのギャップがありそうかを見積もり、そのギャップを埋めるだけの改造コストと、JIS化に対応できなかった場合の業務上の影響度を天秤にかける	将来予測されるシステム変更への対応方針がない	JIS化で影響が生じる範囲を予測できれば、JIS化されても容易に対応できるようにプログラム設計する
18	仕様検討に時間を要し、十分なテストができないまま受け入れテスト（１）	短工期を求められていた上に、仕様検討に時間を要した。仕様の確定した機能から順次開発に着手したが、工期内でテストが十分できていない機能があった。工期を守ることを優先し、その状態でリリースしたため、顧客の受け入れテストで仕様認識違いを含めて不具合が多発した	多少の品質の悪さは、リリース後の受け入れテスト時点での修正でなんとか対応できると考えて、工期を優先してリリースしてしまった	・品質が不十分と思われる部分について、その部分に問題があった場合に手作業や他の機能で業務を遂行でき、かつ、それに対して顧客側が納得しているのであればよい ・顧客の合意を得ないまま、品質の低いシステムを受け入れテストにかけるのは、顧客との関係上、大きな問題に発展することもある	・基本設計、詳細設計の進捗の大幅な遅れ ・顧客の仕様決めの遅れ、仕様変更の多さ	・上流工程であれば、実現のためのケースシナリオ（楽観、悲観、最悪）を決めておき、最悪のケースにおける、スケジュールと最小限の実現機能を顧客と合意しておく ・すでに受け入れテストに入っている場合は、受け入れテストを中断し、スケジュールを延期する。仕様再確認、修正後テストを再実施する。その後、顧客のキーマンによる仕様確認テスト（受け入れてスト前）を実施後、リリースする
19	仕様検討に時間を要し、十分なテストができないまま受け入れテスト（２）	基本設計で仕様確定に時間が掛かり、他に影響する可能性がある機能の詳細設計が遅れていた。その部分の詳細設計が完全に終わらないうちに、出来る部分からプログラミングに着手。しかしその部分のプログラミングで設計工程に起因する問題が多発し、手戻り量が多かった。その対応のために工期を守れなくなった	工期が守れない場合、スケジュールの延期を顧客にお願いすれば、顧客は受け入れてくれる可能性はあると考え、できるものからプログラミングに着手した（結果的には顧客も受け入れてくれた）	・スケジュール延期が許されるだろうというプロジェクトスケジュール管理の問題と、システム構築プロセスの問題を一緒にしてはいけない ・影響範囲が広いシステム機能部分の詳細設計が遅れているときに、できたところから着手してよいと見切ることができるのは、詳細設計の後の変更が着手部分にどの程度の影響が出るかを把握しているときである。インターフェースの改修で先行着手部分が生き残るかどうかである	他に影響する可能性がある機能の詳細設計が遅れる	・上流／下流工程関わらず、工期が遅れることが予想される場合は、カットオーバーのスケジュールの延長についてなるべく早く顧客と協議する ・同様に、仕様の縮小や段階的なリリースを検討する ・効果的であると判断される場合には、要員の追加投入を検討する（ただし闇雲な要員投入はかえって効率を悪くする場合もある）
20	顧客側メンバーの体制が弱く、仕様が詰められない	顧客側メンバーは、システム化の経験が浅く、現場部門に対する仕様決定権も弱いので、仕様確定には計画より期間がかかる可能性があったが、ベンダー側の戦略上受注する必要があった。プロジェクトが始まると、顧客側メンバーが多忙との理由により、仕様検討の稼働や時間が十分に取れず仕様確定が遅れ始めた。顧客側の要件の提出期限をベンダー側から提示。顧客は、遅れをとりもどすためメンバーの増強を行なった。 また、スケジュールを延期することとなり、当初予定していた納期は守れなかった	当初ベンダー側としては、顧客側メンバーのスキル不足を補う形で、ベンダー側で作成した仕様案をもとに打ち合わせすることにより、仕様確定をスムーズに行い計画通りに納めることができると考えて受注した	・顧客側の体制やスキルの不足が事前にわかっているのであれば、顧客側の体制強化を依頼する先（相談相手）を見極めておくこと ・仕様のたたき台が作成できるほどのスキルがベンダー側にあり、現行システムのリプレース案件である場合は、現行システムのドキュメントをベースに仕様詰めする部分を絞るなどの可能性を検討しておくこと ・契約時に顧客側に起因するスケジュール遅延や予算オーバーについての免責事項を入れておくべきである	顧客側メンバーの経験が浅く、決定権も弱い	・顧客側の責任を明確にした上で、顧客側の体制強化を依頼する ・システム検討範囲の縮小や工期延長について協議する ・現行システムのドキュメントなどをもらって、それをベースに検討を進められるかを判断する
21	不条理な予算圧縮要請で開発に着手	システム化の対象領域の業務知識に乏しかったが、受注しないと、この先顧客との取り引きが続かないという危機感があった。請負契約をしてしまったが、大幅な工数増になり、プロジェクトの想定していたよりも採算が大きく悪化した	顧客からの「簡単にできる」という言葉にも反論できず、顧客予算に合わせた見積額の圧縮要請を受け入れて受注し、開発に着手した。赤字になっても受注するほうがよいと判断した	・予算の圧縮要請を受けても、それに見合うだけの前提条件について合意されていること（対処例参照） ・工数増分と原因を把握しておき、都度顧客に説明し、できるかぎり費用追加を認めてもらうことは必要	・対象領域の知見に乏しい ・根拠のない費用圧縮要請	・圧縮要請に対しては、それに見合うだけの前提条件を提示し合意してもらうこと。たとえば「現行システムのドキュメント提示」や「顧客側メンバーの業務知識の提供およびメンバーの積極的な参画」など ・（業務知識が受注側に不足している場合は難しいが）予算に応じた仕様検討対象範囲（開発範囲）の合意 ・仕様検討対象範囲(開発範囲）増に伴う再見積もりと対処策の事前確認 ・予算の増額要求
22	さほど大きな影響はないだろうと思い、既存システムのプログラムを修正	既存システムに対する追加機能の開発を受注した。追加部分の開発は、共通基盤部分にも手を入れる必要があった。ただ、契約範囲外ではあったが、現行システムのメンテナンス性の向上を図るために、自社費用で共通部分のプログラム修正を行うことにした。ところが結合テストでその修正に起因する不具合が多発し、手戻りによる工数増と工期の遅れが生じた。既開発プログラムの修正は、自社理由によるものであり、これによるスケジュール延期を顧客に話すことができなかった	修正による影響個所は少ないと見込んでいたので、修正部分の単体テストを十分実施しないまま、追加機能の結合テストを行った	・状況から察するに、既存部分のテストのし直しが大きな工数になるため、テストのし直しをやらないで済ますことを考えていたと思われる。影響が少ないだろうという楽観的な見切りは行ってはならない。すでに稼働しているシステムに手を入れる場合は「何かが起こる」という危機感を持つこと ・最低限、いつでも現行の稼働状況に戻せるようにしておくこと	そのプロジェクトで、戻し手順と戻しによる影響度について検討されていない場合は、既存分部の改修に対して甘く見ている	・既存改修部分に関わる結合テストを優先して実行し、また追加機能部分ではない別の現行システムについてもテストを実施する ・既存改修部分に対するソースコードレビューの徹底
23	時間のかかる顧客側承認プロセスを見切って、承認前に作業着手	短納期のプロジェクトであったにも関わらず、顧客側の意志決定プロセスがシステム部門・業務部門に関連し、複雑になっており承認までの時間がかかることが多く、顧客側の検討に多くの時間が費やされてしまった。顧客側から提示された仕様が、想定して製造していた仕様と大きく異なってしまい、テスト実施中にもかかわらず大幅に手戻りが発生した。顧客との打合せは週１回で実施され、仕様の確認プロセスに非常に多くの時間が費やされていた	仕様の顧客承認を得るまでに時間がかかりすぎる上に、承認を待ってから着手したのでは納期を守れないと判断したため、承認を得る前に製造に着手した	・正式には顧客側承認を得てから着手するべきだが、顧客側承認を見切るには、最低限、顧客側キーパーソンの仕様に対するコミッションを得ていることである。たとえ顧客側で、その仕様に対する反対意見が出ても、声の大きいキーパーソンであれば、反対を押し切ってでも当初の仕様で押し切ってくれることが多い ・上記の見切りでは、誰がキーパーソンかということをよく把握していることが条件になる	顧客側担当者と仕様案の確定度合いについてヒアリングし、もし「協議の場に出してみないとなんとも言えない」という回答である場合、その仕様がひっくり返される可能性がある（またはそれで押し切ろうという担当者の熟意がない）	・顧客側のキーパーソン（必ずしも担当者とは限らない）を見定めて、仕様に対する内定をもらっておく。できればキーパーソンと思われる人物を複数人押さえておく ・工程（マイルストーン）と先行着手のリスクについて顧客と協議し、リスク発生時の責任分担・役割分担について合意を得ておく ・顧客にプロジェクトのスピード感に応じた体制をとってもらう（線表を後ろから引くと顧客側にも危機感が生まれる） ・仕様(開発範囲）増に伴う再見積もりと対処策の事前確認
24	契約金が未決定ながら進めたプロジェクトだったが、契約時に予算額が大幅に減額	顧客から「契約する」との約束があったが、契約金額についてはまだ未調整の状態であった。案件は短納期でかつ納期厳守のプロジェクトであった。時間もないので作業着手。しかし、契約締結時には想定していた金額よりも受注額が少なかったため、コストが超過し採算が取れなくなった	契約金額が確定されないまま、想定される規模に相当する人員を投入して開発作業に着手した	・契約締結前に作業を開始することがそもそもまずいが、たとえ締結前であったとしても、契約前作業をすることの覚え書きを書いてもらうようにする ・その上で、システムの開発範囲のうち、優先度の高いもの（すなわちシステムのコア部分）を先に開発するなどし、予算を低減されても開発はするだろうという部分に着手する	・金額については別途調整と顧客に保留された ・未契約状態でのプロジェクト立ち上げ	・なかなか契約金額が決まらないという状況である場合、見積有効期限を明確にして、そこまでに契約をしてもらわねば納期が守れない事を明確にする ・これから着手するという段階において、必要最低限の機能範囲に絞り込んで着手するなどのリスク低減を検討する。その際に営業担当者から顧客の予算額についての確度を確認しておく ・作業開始後であれば、要員投入と作業／稼動実績の顧客報告を行い金額交渉を行う

事例番号	事例タイトル	概要	事例における見切りの内容	本来の見切りの考え方(※)	捉えるべき兆候	対処例
25	保守フェーズにおける作業慣習で、仕様変更の見積もりとともに作業を着手してしまった	常日頃から保守SEは顧客からの依頼があったら対応するという慣習があった。ただし今回は、顧客は緊急依頼として、仕様変更を見積もって欲しいと保守SEに依頼した。顧客の都合で、仕様変更は実施しないことになり、仕様変更設計や開発作業が無駄になった	これまでの慣習から、顧客側の承認は後から得られるものと考えて、通常通り開発に着手した	・この事例は慣習にしたがって作業着手したということで、「開発することになるだろう」という実施の見込みをして作業に入っている。本来してはいけない内容 ・やってしまった分は泣くしかないが、開発成果物や検討内容などの情報は保管しておき、次期システム開発等にて回収できるように調整する	・プロジェクト・マネージャが「誰が今何をしているのか」を管理できていない ・保守案件の仕様変更や機能追加などのテーマ管理がされていない ・忙しくないはずなのに、忙しくしているメンバーがいる	合意内容について文書化を行っておき、作業実施記録も詳細に記録しておく
26	最新技術てんこ盛りで、納期も短いシステム構築で手戻り多発	重要顧客の記念事業として、最新技術をふんだんに盛り込んだ社内システムを計画していたが、技術的に困難が予想され、しかも短納期で開発する必要があった。プロトタイピングなどあらかじめ検証する工数や工期も取れなかった。設計～製造フェーズに入り、実現できない機能が多発し、一部は要件定義に手戻りが生じた。結果、工数増大し不採算プロジェクトとなった。	実現性、生産性、性能など未知の領域を残したままだったが、顧客側の押しも強く、なんとかなるだろうと思い開発に着手した	・通常、新技術を熟知した人材を集めることは困難である。人をかき集めればなんとかなるという発想は危険。人員追加投入を繰り返し、不採算プロジェクトとなることも多い ・本来は委任契約と請負契約を使い分け、技術リスクの高い工程は委任契約とすべき。重要な顧客であるなら、なおのこと、実現性の低いプロジェクトに対して「ノー」と言うべきである	引き合い時の顧客の過度の期待	開発が間に合わなかった場合に備えて、既存技術での実装や他パッケージの利用も想定しておく。それに備えて、どのくらいの遅れが発生したら開発方法を切り替えるかを検討し、顧客と合意しておく
27	ＤＢＭＳが詳細設計後に決まって手戻り	実績、機能、価格の異なる2つのDBMSの何れを使うか顧客が迷っていた。また、マルチベンダーでの開発案件だったが、開発ベンダー間でもそれぞれのDBMSを押す勢力ができてしまい、お互いに譲らない形になった。そのため顧客も更に迷い、DBMS決定に手間取った。詳細設計終盤で顧客が、当プロジェクトの想定DBMSとは別のDBMSに決定。一部詳細設計が無駄になった	プロジェクトの特性や世の中のトレンドを考え、一方のDBMSを使うことになるだろうと見切り、詳細設計に着手した	・想定外のDBMSになったときにリカバリが間に合うのであれば、先行着手することも可能 ・そのためには、システムの設計や実装のうち、どの部分がDBMSの製品に依存するかを把握できており、さらにDBMSが想定外にものになった際の回収コストと期間を見積もっておく必要がある。プロジェクトで想定する期間・予算のバッファ（保険）があると思うが、それに収まるかどうかを判断すること	基本設計時のDBMS選定に関するすれ違い	・まだ実装着手前である場合、顧客にDBMSの製品特性が実装方式に与える影響を説明し、早期の決定を促すこと ・実現方式に依存する部分としない部分を明確にして開発を進めるように周知徹底する。もし他のDBMSになった場合を想定して、他DBMS案を押してきたチームに対して技術提供をしてもらう ・別部署へ応援手配し最低限の技術者を確保すること
28	開発効率向上のためのフレームワーク作りがボトルネックになってしまった	ソフトウェア開発フレームの標準化による開発効率性と保守性の向上を図るため、専用のフレームワークを開発し、その上に業務プログラムを乗せることにした。専用フレームワークの設計・開発にアサインした人材の能力が足りず、設計作業がずるずると遅延した。このままでは全ての開発が止まるため、専用フレームワークの開発を止めた。最終的に、フレームワーク機能を縮小し設計作業を進めたが、似たような別モジュールが多く設計されてしまい、開発効率が低下し、工数が増え、工期が遅れることとなった	専用フレームワークの設計作業に、適した人材をアサインすることができず、なんとかなるだろうと思い、別の人材をアサインした	・本プロジェクトでは専用フレームワークの構築がクリティカルパスであり、重要なマネジメントポイントである。アサインした人材が適していたか適していなかったかは結果論であって、本来は重点的なマネジメントを実施して、危険と判断したら人員投入するか、フレームワーク開発そのものを諦めて他のフレームワークを利用するなどの対応策を発動するべきであった ・クリティカルパスが何かをよく考え、そのクリティカルパスに問題が発生した場合の代替案を準備しておくことと、その代替案切り替えの判断基準を決めておくことが重要である。代替案と代替案を発動した場合の影響範囲を把握し、影響度合いがプロジェクト内で処理できるレベルであれば、見切ってもよい	・フレームワーク設計メンバーの能力不足 ・フレームワークの開発ドキュメントのできばえ	・フレームワークの開発が失敗したときのことを考えて、機能縮小の影響度合いや範囲を確認しておき、その範囲について顧客と合意しておく ・フレームワークの開発が失敗した場合の代替案を準備し、代替案発動時の影響度も考慮しておく
29	プロジェクト・マネージャに大規模プロジェクトの経験がなく、結合テストの頃から調整不足が露呈	大規模プロジェクトを受注したが、プロジェクト・マネージャを任命する責任者に大規模プロジェクトの経験がなく、そのプロジェクト・マネージャが適任かどうかを判断する能力がなかった。要件定義から設計、プログラミング、サブシステム内結合テストまでは、計画通りに順調に進んでいるように見えていた。しかし、サブシステム間テストの段階で、結合テストの進め方やテストデータの準備を誰がするのかといった調整不足が露呈し、テスト作業が停滞してしまった	責任者が、大規模プロジェクトをマネジメントできるスキルを確認せずに、プロジェクト・マネージャを任命した	・大規模システムかどうかに限らず、プロジェクト・マネージャがそのプロジェクトを制御できるかどうかを、プロジェクト・マネージャ任命者は見極めなければならない。少なくとも、そのプロジェクト・マネージャのスキル不足による問題が発生することも考えて、プロジェクト支援のための組織（PMOなど）をバックアップに付けておく ・後進を育てる意味もあるが、そのプロジェクト・マネージャが倒れた場合もしくは機能しない場合を想定して、有能な若手をサブリーダに付けておけば、プロジェクト・マネージャ交代時のリカバリも早い	・プロジェクト・マネージャ自身がリスクとして認識していないので、不調の予測は難しい ・上流工程の段階で、下流工程での作業内容やリスクについて想定していない（想定できない）場合は結合テストの頃からまごつく可能性あり	・組織として、高い頻度でプロジェクト実行状況のレビューをして、プロジェクト・マネージャが適任でないことを早く見つける ・一刻も早く、必要なスキルをもったプロジェクト・マネージャを投入する
30	未知のパッケージを協力会社から提案してもらったが、そのパッケージに品質問題発生	顧客に対する提案内容として、自社内では、ある中核機能に関する実現手段を持っていなかったため、協力会社に提案してもらったパッケージを利用する方法を採用した。しかし、いざその方法で構築作業に入ったところ、提案されたパッケージに重大な品質不良が発覚した。提案してきた協力会社配下にて、さらにパッケージベンダーとハードウェアベンダーの間で、原因の切り分けができず、対策に時間と費用を要した	中核機能の実装にあたって、協力会社がパッケージを提案してきたが、プロジェクト・マネージャにはパッケージの内容については理解不足で判断しようもなく、協力会社を信じてパッケージの品質には問題がないだろうと判断した	・機能実現にあたっての重要な機能および性能についてリストアップしておき、その部分についてのパッケージの品質を確認しておく必要がある ・そのパッケージが実は開発中のベータ版しかなく、品質確認が難しい場合は、少なくとも代替運用か、または代替パッケージを用意することによる抜け道を作っておく必要がある	パッケージまたはソリューションの品質または性能に関して、ベンダー側のコメントがない。もしくは品質・性能の根拠を示さない	・必要最小限の機能、性能で取りあえず「リリースする。その後、性能向上を図る ・パッケージベンダーに機能保証をしてもらうとともに、トラブルへの迅速対応のために多階層の関係をフラットにするようリーダを決めておく
31	経験の少ないオープン系システムで、プロジェクト・マネージャがコントロール不能に	未経験のオープン・システム基盤を用いたシステム開発案件で、顧客にとって非常に重要であり、セキュリティ要件や性能要件など、コントロールの難しい要素が多数あるプロジェクトであった。ベンダーの経験不足・知識不足もあり、安易な体制でスタートしたため、プロジェクト・マネージャは、あがって来る懸案事項や日々のタスクを処理しきれない状態になり、統括できなくなってしまった	受注前に十分調査をせず、マネジメントの問題が発生するだろうと想定することもなく、安易なプロジェクト体制で受注した	・新しい開発方法論や技術などを利用する場合、どのようなことが起こるかを想定することが難しいため、リスクも多く、問題となって顕在化する可能性も多い ・基本設計の段階で専門家などの有識者によるレビューをするなどの手段を講じることができるのであれば、軟着陸できる可能性はある	要件の確定が遅れる（業務改善検討結果が影響している）	基本設計の段階からオープンシステム基盤の専門家など有識者の投入をする
32	複数会社でのプロジェクトで旗振り役が不在	ポータルサイトの実証実験に複数社が参加し、プロジェクト全体としての実験項目だけが決められていた。実証実験システムの開発プロジェクトの全体をとりまとめる体制がなく旗振り役がいなかったため、発注側が意図している実験項目が消化できないほどシステム開発が遅れた。システム開発と実証実験のスケジュール調整もできていなかった。各ベンダーでのプロジェクトコントロールがきかなくなってしまった	発注者側は、各ベンダーで意図するとおりの役割分担になっているだろうと見切ってスタートした（ベンダーの立場から考えると自社の範囲分についてあらかじめ与えられた予算内で一定の成果さえ達成すればよいので、全体的な管理を考える必要性はなかった）	・役割分担や開発範囲については齟齬が起きるのが当然であるという意識を持つべきで、発注者側はオーナーシップを発揮しなければならない ・「うまいことやってくれるだろう」という淡い期待だけで、プロジェクトの成否に関わるスコーピングを見切ってはいけない	・要件を確定するのが誰かが曖昧である場合、プロジェクトのオーナーが実質不在であると考えてよい ・お互いが領空侵犯しないように気を遣っている、もしくは各々の事情で情報公開をあまりしない傾向がある場合は、開発範囲で抜け漏れにつながる可能性がある	・発注側による積極的なプロジェクト参画は必須 ・発注側と受注側の複数社それぞれからマネージャ／リーダクラスの代表者を選出してもらい、意志決定を行うための合同会議体を作る ・または、取りまとめ役のベンダーを指定して、リーダーシップを発揮してもらうようお願いする ・すでに不調に陥っている場合、現状調査を行って課題の洗い出しを行い、担当社を取り決めるなどの旗振りを発注側主導で進める

事例番号	事例タイトル	概要	事例における見切りの内容	本来の見切りの考え方(※)	捉えるべき兆候	対処例
33	顧客指定のフレームワークで受注し、大幅コスト増	顧客からの提案要請時に、アプリケーションフレームワーク（FW）を利用しての開発が前提になっていた。開発を進めるうちに、FWが機能的に不十分であることが分かり、FWの上に載せるアプリケーション側で機能を補完することになった。FWの品質や性能にも問題があり、アプリケーション開発部分のテスト工程で、障害が多発したり性能がでないなどの問題もあり、ハードウェア増強など大幅なコスト増大となった	FWがターゲット業務に適合するかどうか、実現可能なのかを吟味せずにFWを適用することが決定された	・FWの適用に関しては、想定している機能に関する性能や品質について見通しが立っているかを考える必要がある ・その見通しが立てられない場合、そのFWが全く使えないことを想定して作り直す場合や他のFWを持ってくる場合の工数を見積もっておき、線表を後ろから引いて、“顧客指定FWの品質を判断するエンドポイント”を決め、それまでにFWを評価できるかを考える ・FWの利用はもともと効率性を考えた上での戦略なので、上記のように代替案で後ろから線表を引くというのは現実的ではないことが多い（代替案のほうが期間がかかることが多い）。この場合、プロジェクトとしては顧客指定FWに問題があった場合の免責事項を契約条件の中に含めることでリスクヘッジする必要がある	・顧客側にそのFWを実地に利用した経験がない ・そのFWの評判がよくない ・FWそのものが新製品である、または、導入実績が少ない ・FWについて性能評価結果が存在しない	・上流工程においては、なるべく早期にFWの品質検証をする期間を設けることを計画する。同じFWを使った他のシステムやプロジェクトを調査するとよい ・また、FWの品質・性能に問題がありそうな場合を想定して、顧客との合意の上で、代替FWを用意する手はずも整えておく ・体制強化として、その顧客指定FWの経験者を開発メンバに加えたり、FW開発者と技術支援契約を結ぶ ・下流工程に入ってしまっている場合は、事例にあるようにHW増強をしたり、ボトルネックを探るなどの地道な性能改善をするしかない
34	本番データに現場部門のパッチが当たっており、論理的不整合が多く移行コスト増大	長年使用してきた現行システムの再構築案件。現行システムのデータを、正確に新システムに移行することが業務上必須であった。ところがいざプロジェクトを開始したところ、売掛残（債権管理）システムについて、現行の仕様が不明確であるためデータ移行ロジックを正しく作れないことが判明。しかも移行すべきデータに、現場部門で手作業でデータ修正を加えて運用していることに気づいた。結果として、データ移行に多大な費用と期間がかかり、カットオーバー後も業務上の障害が発生しつづけた	提供してもらう予定の現行システムの仕様書が正確で、かつ移行するデータが論理的にも整合しているはずと考えてスケジュールと費用を算出してプロジェクト計画とした	・現行システム踏襲型のプロジェクトでは、現行システムに関する情報がどれだけ提供してもらえるかがプロジェクトの成否を握る。したがって、現行システムに関する仕様書などを見て、どのくらいの作業量があるかを見積もった上で、スケジュールと費用を算出する必要がある ・現行データの現場部門での手作業での修正（いわゆる“パッチ”）まではなかなか想定できない。現行システムに入っている元データに論理的な整合性がない以上はプログラムでの移行は困難。致し方ない事例 ・移行ツールを設計するにあたっての前提条件を明確にしておいて、データに問題があったり仕様書などの十分な情報がなかったりした場合の再スケジュールなど、調整余地を残すように顧客と合意しておくこと	移行データの詳細仕様が、催促を繰り返してもなかなか現システムの担当者から提示されない	・プロジェクトの前提条件に現行システムの仕様書の提供やデータ要件、運用状況等についての情報提供をしてもらうことと、それらに不備があった場合には再スケジュールや再見積もりをすることを明記する ・すでに事例のような問題になっている場合は、移行データの手作業パッチの記録や不整合パターンの洗い出しを行う ・移行データの状況判定プログラムを作成し、移行対象データを徹底的に調査する
35	現行システムを再利用した新システムを導入しようとして例外処理に苦しむ	長年利用して保守が困難になっている複数の現行オンラインシステムがあり、利便性向上のためにフロントエンドに新しくワークフローサーバーを置いて、新しい業務を、ワークフローサーバーによるシステム間連携で実現するというのが顧客の要望だった。ところが現行システムの内容を知っている技術者は居らず（そのベンダーはすでに存在していなかった）、現行システムとの結合テストでは例外処理など予想と異なる動きが多発。テストが進まなくなった	プロジェクト体制に、現行システムの内容を熟知している技術者を参画させることが出来るかどうかを確認せず、プロジェクトを計画した。現行システムに関する技術者がいる、もしくは、システムの仕様が明確であろうと思って進めてしまった	・現行システムに関する仕様が明確であることが前提条件であることはこの事例の中にも現れている。この仕様が明確である、もしくは熟知した技術者をプロジェクト体制に組み込むことができることが前提条件であるので、これらの前提条件が覆されたときの対応処置案がきちんとあれば、プロジェクトを進めてもよい ・実現可能性について懸念がある場合は、実現するレベルを抑えたり（たとえば機能を絞るとか、運用方法を利用側で注意するなど）、最悪の場合はプロジェクトを中止するなどの判断基準を顧客と取り決めておく、といった対応策を準備しておく	・見積もり時から、現行オンラインシステムの仕様が十分提示されていない（特に入力エラー時などの障害処理） ・現行システムが非常に古いシステムである（技術者や仕様を決めた者がすでにいない可能性が高い）	・まだ上流工程であれば、実現可能性について検討し、実現にあたっての課題出しから始める。そのために、現行システムの調査をしてから、新システムの実現方針を考える ・下流工程ですでに事例のような不調に陥っている場合、実現レベル（実現機能や運用制限など）について顧客と合意した上で、スコープを決めて開発に入るか、もしくはプロジェクトを中止する ・例外処理や処理の組み合わせパターンを把握できないなど、今後の作業量が見えないうちは、開発作業に入らないようにすること
36	顧客側からの要件出しが細切れで、なかなか要件確定してくれない	教育機関の顧客で、他社が開発した旧システムをリプレースする案件。開発担当社側には業務ノウハウが無かった。要件定義工程では、顧客からの仕様提示が細切れで、なかなか要件が確定しない。プロトタイピングしながら進めたが顧客は否認を繰り返すばかりで大幅な納期遅延が生じた。ついには要件定義もそこそこで設計に入ったが手戻りが多発。プログラムの品質も悪く、開発要員を多数投入したが納期遅延。要員も疲弊し、デスマープロジェクトとなった	受注側は顧客側のノウハウ不足のため仕様が確定できないのだと判断し、プロトタイピングとして一部業務について仕様が確定しないまま先行開発を行なった	・システムのプロトタイプを見せることによって顧客側の新業務内容検討を促進させることができる場合もあるが、それは受注側に豊富な業務知識・経験があり、それをベースに適切なプロトタイプを提供できるときである。勝手なイメージだけでプロトタイプを作っても、全く別の業務処理が策定されて、そのプロトタイプが無に帰すことになる可能性が高い。事例のように、顧客に納得してもらえない場合も多い ・顧客側から「こういうイメージのプロトタイプを作って欲しい」という要請があった場合を含め、プロトタイプの内容について顧客側と合意が取れていればプロトタイプを進める価値はある。いずれにしてもプロトタイプの効果を考えてから開発に進むこと	・客先からの仕様提示が細切れ、曖昧である ・客先の意思決定の責任が分散しており、キーパーソンが不明	・顧客側に構築期間に対する危機感がない場合は、仕様凍結の時期を明確に宣言し、顧客側で早急に決めてもらう必要があることを理解してもらう ・顧客側に要件を決める能力が不足していると思われる場合は、要件定義の前取りをする。検討体制や意志決定プロセスの定義・確認からシステム化対象範囲の確定、検討事項の整理などを進める。そのための上流工程要員を体制に新たに組み入れるなどの対応を進める
37	既存資産が流用できると思っていたが、流用できないことが分かりコストオーバー	ある地方自治体のシステム化案件。他の自治体で、同じ業務処理をするシステム案件を実施したことがあったので、そのときの成果を流用できると判断して見積もりをし、開発を着手した。ところが、開発を進めていくうちに、業務処理が想定と大きく異なり、既存のシステムは流用できないことが判明。結果として大幅に開発規模の増大し、開発期間も長期化した	類似システムが流用可能と考えて、それを前提に原価を低めに見積もり、プロジェクトを計画した	・類似システムを流用できるだろうと見込んで開発に着手し、当初想定規模を大きく上回る開発規模になるということはよくあることであり、既存資産を流用することを前提とする計画立案時には注意が必要である ・流用部分なしで新規開発する想定で見積もりを出すと、他のベンダに価格面で負けてしまうなどの営業的な要素も絡むため、流用の適用／非適用の判断はプロジェクト・マネージャだけでできるものではない。既存資産の流用がプロジェクトの前提であるなら、既存資産の流用をすることと流用が適わなかった場合の費用・納期の変更について顧客側と同意を取ること	・費用削減だけを目的とした既存資産流用 ・業務名、システム名だけで既存資産流用可能と判断する見通しの甘さ	・流用しない場合の見積もりと流用した場合の見積もりを顧客側に提示し、流用が適わなかった場合の費用追加・納期の延期について合意を得ておく ・流用部分についての適用可能性について評価する工程を追加し、その評価結果に応じたその後の進め方について事前に検討しておく
38	顧客との共同開発という名目で、イニシアチブが取れずに要件がなかなか確定できない	医療法人である顧客と、開発受託会社とで費用折半でシステムを共同研究・開発することになった。開発したシステムは、開発受託会社で他の医療法人へのパッケージ展開する計画で顧客とも合意していたが、その顧客にしか適用できないような個別要件が噴出し、歯止めが利かず、設計手直しなどの手戻りが多発。共同開発ということで、開発側もイニシアチブを発揮することができなかった。結果、ローカル仕様ばかりの、他社展開できないようなシステムができあがった	顧客側の仕様確定が遅延していたが、共同研究のメリットを重視し、仕様未確定であるにもかかわらず開発を開始した	・共同研究や共同開発の場合、顧客側と受託側とで、役割分担が曖昧のままプロジェクトが進行することが多い。業務ノウハウをなるべく引き出して他社展開できるパッケージを作りたいというシステム開発側と、たくさんの要件を入れ込んで自分たちにとって使いやすいシステムが欲しいという顧客側との思惑が一致して、要件が山のように膨れるが、金と納期が決まっているので「どこまでやるのか」という仕切りが必要になる ・全体の要件は未確定でも、確定部分と未確定部分の整理ができていれば、確定部分を先行着手することは可能	・共同研究という名目のもと、顧客と開発担当会社との役割分担や責任所在が明確になっていない ・パッケージ化対象機能かローカル対応機能かの判断基準がない、もしくは曖昧	・要件の確定部分と未確定部分の切り分けと、未確定部分による全体への影響度を調査する ・未確定部分による影響が多少あっても先行開発のほうがメリットがあると判断したら着手する。万が一のリカバリ手段も用意しておく ・パッケージ部分への機能取り込みか、ローカル開発部分への機能取り込みかの判断基準を明確にし、ローカル開発部分は追加発注であることを宣言する
39	何でも言うことを聞くという応札条件の案件	医療法人の顧客で、顧客要望を全て取り入れるということが応札条件。顧客要望が膨大に発生し、仕様を取りまとめるはずの客先窓口が破綻したため、要件発散。パッケージのカスタマイズを設計したところ、提案したパッケージでは品質・納期を満たせないことが判明し、別パッケージに切り替えて開発しなおしたため、ニ重開発となり、費用が想定より大幅に増大してしまった	既存パッケージソフトを流用し、一部カスタマイズすることにより、顧客要望が満たせると判断して応札した	・開発範囲と作業期間が決まらないと非定型業務であるプロジェクトを定義し、完了させることができない。したがって、「全ての要求を取り入れる」という終了条件を定義できないような前提条件については、開発範囲と作業期間の制限から顧客側にある程度の要求の押さえ込みを覚悟してもらう必要がある ・プロジェクトのスコープとして受け入れられる要件の量に限度があることを理解してもらい、開発対象範囲を絞ること、その後の要件については追加案件ということで別途費用をもらうなどの合意を少なくとも取ること	全ての要求を聞くというプロジェクトの完了条件が曖昧な応札仕様	・応札の見直し ・応札してしまったとしたら、納期について合意し、用意できる開発要員のリソースから開発可能な規模を算定する。それに見合うような要件までを開発するということを合意してもらう。開発範囲をなんとかして決める必要がある ・最悪のケースとして、ペナルティを払ってでもプロジェクト中止・撤退
40	パッケージの品質と機能は十分に確認しよう	パッケージを利用したシステム開発。開発途中で、パッケージの品質不良が発生し、当初予定した機能の開放が遅れ、システム改修や機能追加が多量に発生した	パッケージの品質・機能・性能について、大きな問題は発生しないだろうという判断した	・パッケージ、特に新規に使うパッケージの場合は、品質、機能を十分に評価して、適用の可否を判断することが重要である ・リスクが高いが、戦略的な理由で、パッケージを使う必要がある場合は、パッケージ部分を先行する、代替案を考慮しておくなどのリスクヘッジが重要である	利用するパッケージの品質、性能を確認せずに、また、利用する機能の提供スケジュールの担保をとらず開発を進めた	パッケージ側の体制強化により品質強化を図る。体制を強化し、再スケジュールする

事例番号	事例タイトル	概要	事例における見切りの内容	本来の見切りの考え方(※)	捉えるべき兆候	対処例
41	仕様変更に関する扱いは、顧客と意識合わせをしておくこと	全国の拠点に導入するシステム開発。ハードウェアも含めSIとして受注。ハードウェアの納入遅れにより、サービスインの遅れが発生。また仕様追加、変更についいても、顧客と交渉したが、有償対応が認められなかった。全国の拠点に導入する支援作業も、顧客との作業規模、範囲に意識違いが発覚し、大幅に増えた	仕様追加の場合の再見積もりなどの取決め、支援作業の内容と範囲、ハードウェア調達遅れの取決め、などを明確にしないまま、開発を進めた	ハードウェアの納期、仕様変更、付帯・支援作業は、納期やコストに与える影響が大きいので、あらかじめ顧客と調整しておく	仕様追加、変更時の取決め、支援作業の具体的な仕切りがないまま、開発が進んだ	・仕様変更の取り扱い、作業支援範囲の再調整 ・サービスインの時期調整 ・費用の扱いを明確にしておく
42	大幅な再利用を前提とした開発は要注意	他システムの再利用80%を前提にシステム開発を計画。プロジェクトの独自仕様、業務プロセスの違い、仕様の増加策に伴い、再利用率が大幅に低下し、開発規模が増大した。 また、仕様確定が遅れた部分に先行着手した結果、手戻りが発生した	他システムと類似しているため、かなりの部分を再利用できると考え、見積もりを実施した	再利用を前提とした業務システムの開発は、パッケージ適用の場合と同様に、業務プロセス、要求仕様を十分に明確にした上で、適用可否の判断をする必要がある。もし、適用を決めた後で、適用できない部分が多いことが判明した場合は、再利用できる範囲が大幅に減少するため、納期遅延と大幅なコスト増になるからである	再利用可能な部分を検証せず、契約した	・体制の増強、管理体制の強化、再スケジュールの実施 ・パッケージの利用率が変わったときの扱いを明確にする
43	作りながら仕様を確認	ノウハウの無い業界相手にパッケージ製品を導入し、仕様の詳細は顧客にヒアリングしながら開発していく計画を立てていた。しかし、ヒアリングを進めると見た目や機能名からは想像できないような仕様が続々明確化され「無いと業務にならない」「常識だ」という言葉に踊らされて開発規模が増大。納期遅延を引き起こした	知らない事を自分の都合の良い方に考え「仕様は不明確だがベースとなるパッケージが必要な部分を充足してくれているだろう」と期待してしまった	・分からない部分を明確にするか、制限とする事で仕切りを作る必要がある ・システムの使用者は理想を語るので、聞いているうちに仕様が膨らんでいく事はよくある。ヒアリング結果と現実を比較する（パッケージ機能とヒアリング内容のフィット＆ギャップを取る）作業と、やる／やらないの検討を別にする必要がある	・「よく分からないけど大丈夫」と思ってしまった。もしくはそう説明された ・ユーザの提示する仕様の反映で仕様検討が遅延し始めた（仕様が明確にならない場合も、後で噴出する可能性がある） ・パッケージのカスタマイズ	・業務を良く知る人員の投入 ・仕様確定に向けた不明確部の明確化と期限管理強化 ・品質確保に向けた変更管理体制の強化 ・パッケージの拡張の扱いを明確にする（戦略案件等）
44	知らない物で決まっていない物を作る	運用開始の日程が確定している状態で仕様検討から開発を開始した。仕様確定が遅延し、納期が迫っている状態であったにも関わらず、未経験の新技術を採用して難度の高い開発を行った。その結果、開発効率が落ちた上に突貫工事で発生する不具が増加し、稼働品質を確保する為に納期遅延を引き起こした	残期間から逆算した計画を立てて対策を打つ必要があったが、新技術採用のリスクバッファを踏まえて「そうは言ってもこれくらいの期間があれば出来るだろう」と甘く見てしまった	・納期ありきで計画を立てる場合には、必ずスケジュールは逆算して検討する必要がある ・未確定な要素（新分野開拓・新技術採用等）は、悪い方で検討し、常にアンテナを高くし、コンティンジェンシープランを立てておく必要がある ・仕様検討という発注側意思が大きく関わる部分でスケジュールに影響が出る物は、その影響を明確にして発注側の責任を明確にする必要がある（製造側にも説明責任がある）	・仕様検討で工程遅延が発生した ・工程分割（リスク軽減）した受注になっていない	・技術ノウハウを持った要員の投入 ・仕様変更と取り込むべき仕様の明確化と管理強化 ・要求を出してくるユーザとの意識差解消と、顧客側トップへ影響把握の申し入れ
45	無理な見積を通すと作り手か使い手のどちらかが破綻する	顧客予算に合わせ強引に原価低減した提案を通した。パッケージ機能に可能な限り準拠した仕様とする事を前提として受注したが、曖昧な線引きの為、カスタマイズが必要となる要求が大量に発生。仕様を確定できないまま原価山積みに合わせて開発を併走させ、手戻りも多発。最後は納期遅延となりペナルティが発生しまった	「パッケージ機能を前提とする」という言葉で改修量を制限できると考え、改修量や内容の閾値を決めていなかった	・前提条件が変わる場合には、その影響を明確にして示す必要がある ・開発範囲の明確化と意識合わせを前提とした上で仕様の調整に入る必要がある	・予算に合わせて受注額を決めている ・前提条件が破綻した ・パッケージのカスタマイズ ・仕様確定に遅延が発生した	・顧客承認が無ければ先に進まない部分を一括範囲に含む場合は、そのリスクと影響を顧客にも理解してもらった上で話を進める ・開発範囲の再確認と徹底 ・範囲外作業の有償化 ・顧客との合意を徹底 ・前提条件が変わった時の扱いの明確化
46	個別に検討したものを繋げてみたら、繋がらなかった	大規模なシステム開発を分割して数社に、それぞれ仕様検討から発注した。個別に開発して出来上がったシステムで業務全体を通すテストを行ったところ、システム全体で仕様の整合性が取れていない状態になっていた。仕様整合性の再検証から計画を立て直し、予算大幅超過の上、納期遅延で顧客に迷惑をかけてしまった	一次請けとして全体を管理しなければならなかったのだが、分配して各社に任せた時点で「個別の機能が出来上がれば全体も出来るだろう」と考えてシステム全体としての取り纏めを放棄してしまった	・システムを稼働させる視点を持ってプロジェクトを見た上で、各社に開発を依頼する必要がある ・取り纏め者として各社間インタフェースや全体仕様など、隙間を埋めたり土台を作ったりする作業が必要である ・顧客と各社の仕様調整については、全体影響がある内容と個別で済む内容の切り分けと調整をする必要がある	・システム全体を説明できる人員がいない ・稼働責任者が明確になっていない ・各社間コミュニケーションの場が無い ・顧客と開発側のレビューワーが機能ごとに違っている	・システム全体仕様の再確認 ・管理体制の強化 ・稼働時期調整と、それに向けたリスタート ・PMOなど、監査組織の設立
47	顧客側の体制、顧客との役割分担、作業分担を明確にすること	顧客が指定する独自ミドルウェアを利用したシステム開発で、業務やその独自ミドルウェアについて十分な知識・スキルがない体制で開始。顧客側は、現行業務の分析やミドルウェアの情報開示に協力的ではなく、上流工程で大幅に遅れが発生し、後工程でも、手戻りが多く発生して納期とコストが超過した	ベンダー側は、顧客側がリーダーシップをもって現行業務分析の実施や独自ミドルの詳細情報開示について実施してくれるだろうと思っていた	顧客側提供のソフトウェアを利用した開発の場合は、請負ではなく委任での契約が望ましいが、請負の場合には、開示条件、ドキュメントのレベル、開発におけるサポートレベルなどの条件をあらかじめ明確にして契約すべきである。また、業務分析においても、顧客側の体制や、役割、作業分担を調整しておく必要がある	独自ミドルウェアの利用に関して、守秘義務の問題から、受注後の開示とされていた。受注後に顧客側ミドルウェア担当者の説明を聞いたが、仕様書の説明を淡々と実施し、あまり踏み込んだ説明をしてくれない。 情報システム部門が業務をよく把握しておらず、帳票一覧も持っていない	・顧客が参画する仕様確認会議の実施 ・プロジェクト要員強化 ・日次会議、および週次（休日）の進捗確認会議実施でプロジェクト管理を強化。全インタフェースの再精査、主要業務評価の前倒し実施 ・専任要員(PMO)による調達費用精査と削減交渉実施 ・経営層への報告実施
48	オフショア開発では、ドキュメントの記述は詳細に、プロジェクト管理はしっかりと	海外ソフトハウスへのオフショアを活用したシステム開発。オフショアを委託する段階では、最近力を付けてきているものの、まだ設計できるほどのレベルにまで落ちていなかった。そのため、海外ソフトハウス側でも勝手な思い込みで作り込み続けた。プロジェクト側も、スキル不足で上がってくる中間成果物を十分に把握できず、システム規模が増え、作業遅れが発生した	顧客側の業務ノウハウについて、ある程度ヒアリングを実施。システムの大まかな姿を概要設計として示し、海外ソフトハウスに委託した	国内発注の場合は、概要レベルの記述でも、スキルがある会社であれば、問題はないが、オフショアの場合は、難しく、より詳細なドキュメントにする必要がある。また、途中成果物を定期的にチェックし問題がないか確認するなど、きめ細かいプロジェクト管理が必要である	海外ソフトハウスに渡したドキュメントが概要レベルにとどまっている。プロジェクト経験の浅いメンバーばかりになっている	体制の増強、管理体制の強化、再スケジュールの実施
49	顧客の体制が弱いときは、仕様追加、変更が多発する可能性が大	業務的にシステム化範囲が広いシステムの開発を受注。発注側の仕切り能力が弱く、業務内容の定義がきちんとできないことが懸念された。それぞれのイテレーションごとに追加要求や仕様変更が頻発し、規模がどんどん膨れた。進捗も遅れだし、品質にも問題が出始めた。規模の増大に伴って試験規模も大幅に必要となり、結果的に工数が大幅超過となってしまった	顧客要件をきちんと抽出することで構築後に大幅な手戻りをなくすことができるだろうと考え、4回のイテレーションを回すスパイラル型で開発を進めることにした	顧客の体制が弱い上に、業務内容も定義されていない状態では、顧客要件を抽出し、仕様を固めるのは非常に難しく、後で仕様追加、変更が多発する可能性が大きい。このような状態で、開発しながら仕様を決めていく開発手法は非常にリスクが高い	プロジェクトの全体像が見えていない。スケジュールの関係から、仕様が十分に決まらない状態で、設計、製造に着手する	体制の増強、管理体制の強化、リスケジュールの実施
50	規模が大きくなくても、協力会社に丸投げは危険	中規模のシステム開発。協力会社は4社へ発注。その中の1社において、設計不良から、製造に大幅な手戻りが発生。要員を大量に投入してリカバリを図るが、品質問題が頻発。その結果、プロジェクト全体が遅延し始めた。体制を強化し、マネジメントの強化と品質向上を図ってリカバリしたが、費用は大幅に超過してしまった	予算の関係上、社員1名に協力会社4社を付けて対応することにした	・中小規模の開発においても、社員1人で4社の外部委託管理を行うのは、外部委託先への丸投げと等しい ・問題を早期発見し、適切なタイミングで対処できず、問題が大きくなる可能性がある	1人で多数の協力会社をマネジメントしている。管理体制が弱い	体制の増強、管理体制の強化

事例番号	事例タイトル	概要	事例における見切りの内容	本来の見切りの考え方(※)	捉えるべき兆候	対処例
51	信頼関係だけでは適切な課題管理はできない	SIベンダーが受注したシステム開発に際し、要件定義をSIベンダーが実施した後、基本設計以降の作業を実績がある協力会社に全面委託した。信頼関係を根拠に、設計内容の品質を進捗報告だけでフォローした結果、協力会社だけでは解決できない重大課題への対処が遅れ、上流工程の完了が大幅に遅延することとなった	要件確認をSIベンダーが行ったことで当該プロジェクトに係る主要な要件は見切れたとの認識から、基本設計以降の作業を協力会社に全面委託した。加えて、当該協力会社の実力をこれまでの実績から評価していたこともあり、作業の進捗報告だけで管理を行い、個々の課題管理等を共有しなかった	・協力会社に設計を委託する場合、設計品質の可視化と課題の達成状況の管理がSIベンダーにとって重要なタスクとなる ・また、新規業務の開発を委託した場合などは、協力会社が抽出する課題やWBSなどの網羅性を捕捉できるSIベンダーの体制作りが、プロジェクト・マネージャの重要なタスクであることを理解しておきたい	・SIベンダー側で、結合テスト・統合テストの仕様書が書けない ・進捗が計画どおりに進捗している	・SIベンダーが求める設計水準や設計範囲などを協力会社と共有する目的で、当該SIベンダー側で設計書記載のガイドラインを事前に作成し、協力会社との間で充分なレビューを行っておく ・進捗報告以外に、設計内容の段階的なレビュー会を適宜実施することで、設計内容の網羅性や品質についてタイムリーにフォローする ・設計作業の過程で発生する課題のエスカレーションルールを協力会社との間で確立する ・協力会社との契約状況再検証
52	インパクトが大きい要件への対処は先送りしない	要件定義完了段階で、RFPおよび提案書に記載しないような粒度の機能を新規要件として追加した際、当該システムの開発に不可欠な基本機能との認識から、コスト増に関する顧客側の承諾が得られなかった。交渉の長期化に伴う次工程への影響を配慮し、新規要件にかかわるコストをプロジェクトの持ち出しとせざるを得ない結果を招いた	提案時に機能単位の見積もりを要求されなかったこともあり、要件定義の段階でも、個々の機能に関するコストのインパクトや顧客側のコスト感に関する調整、確認を顧客との間で実施しないまま、要件定義の完了時点でコストの見積もりを一括で提出してしまった	・顧客側の予算枠やコスト増への手続きなどについては、可能な範囲で、営業を通じて事前に把握しておくことで、要件定義または基本設計段階で新規要件が発生する都度、顧客担当者と事前に相談すべき案件規模の見極めやタイミングをPM自身が判断することができるようになる ・顧客側では見えにくいシステム機能に係る要件については、要件定義や基本設計の内容をフィックスする以前に、提案内容との関連やコストへのインパクトについて顧客に説明をする機会を設けることで、見積もり範囲の妥当性に関する合意形成が容易になる	・要件定義の内容には関心があるが、コストについて不安を感じていそうもない雰囲気 ・業務に関する要件定義には的確に応じてくれるが、システム面の見識には乏しそうな顧客側の開発体制	・提案時に、顧客側から特段の要求がなくても、RFPに記載されている要件から推定される実装機能の種類や規模とコストとの関連を見積もりの添付資料として顧客側に提出する ・見積もりコストにインパクトのある新規要件が発生する都度、コストや納期への影響にフォーカスした意識合わせを、営業や顧客との間で実施する
53	フィット＆ギャップ分析をせずにパッケージ導入の決断はしない	短期開発という事情もあり、業務要件とパッケージ仕様のフィット＆ギャップ分析を実施せずに上流工程を完了したが、エンドユーザによるテストで、パッケージソフトでは対応が困難な業務が存在することが判明し、一部の機能に関し要件定義のやり直しを迫られ、大幅な遅延とコスト負担をもたらす結果となった	提案時に既存システムの画面や帳票レイアウトを確認した範囲では、過去に発生した法制度や規則等の変更に伴う複雑な業務仕様を想定できないまま、開発コスト低減を目的にパッケージソフトの導入を提案した。顧客側が既存ベンダーに細かい業務仕様の検討を丸投げしていたこともあり、要件確認段階でシステム部門、ユーザ部門ともに"現行保証"以上の要件提示がなかったため、フィット＆ギャップ分析に実施しないまま、パッケージソフトによる開発を決定してしまった	・"現行保証"という業務要件だけで行なうシステム開発は、顧客側でどこまで要件が見えているのか把握しづらい。上流工程の早い段階で、業務の複雑性や特異性を把握しているキーパーソン（多くの場合、既存ベンダーである可能性が高い）を見極め、業務要件にかかわる難易度をテーマにインタビューを重ねることで、パッケージソフト適用の是非を早期に見切ることが容易となる ・上流工程では見えにくい仕様追加が発生する可能性がある場合は、導入するパッケージソフトのカスタマイズ機能や開発体制などの詳細を事前に押さえておくことで、問題発生時の打ち手を幅広に持っておくことができる	・顧客側のユーザ部門と打ち合わせをしても、過去の苦労話が一切出てこない ・要件確認を行う際、システム部門との関係が悪そうにみえないのに、なぜかユーザ部門の参加率が悪い（業務を担当するユーザ部門も、要件定義で話せる内容が少ない）	・始めて扱う業務開発に関しては、過去の開発経緯などを把握している既存ベンダーを開発体制に組み入れるよう、営業などを通じて早期に調整する ・要件定義の早い段階で、受注側が把握している業務要件に関して顧客側のユーザ部門を含めたレビューを実施し、パッケージソフトを提案した妥当性について、顧客側の理解を求めておく必要がある ・パッケージソフトのサポート範囲（パッケージソフトでできること）は、上流工程の早い段階で顧客側に説明を行い、想定外の仕様追加への対応策について、コスト負担を含め、顧客側の了解を取り付けておく ・経営、営業部門含めた課題管理の実施
54	過去データ移行の難易度はプロジェクト・コストにインパクトあり	他社リプレースに際し、過去データの量的規模を見極めたうえでデータ移行に係る見積もりを行った。だが、過去に幾度となく実施された制度変更を踏まえたデータの連続性が、帳票作成やデータ検索処理に求められることを把握し切れなかった。請負契約のなか、要件実現に必要な移行ツールやデータ変換機能などの開発コストを受注側で捻出する結果となった	過去データのなかにはオペレーションミスなどによる不正なデータが一部含まれている可能性については覚悟していたが、過去の制度変更に伴うデータ移行や業務処理などへの配慮といった特殊な要件が見えないまま、設計工程を完了してしまった	・過去データの取り扱いは、顧客側にも見えにくい要件の一つである ・また、制度変更時にビジネスロジックで対応することで、実際のデータには手を付けなかったケースも意外と多い。データ移行に関する具体的な要件を把握しづらい場合は、データ移行の特殊性を意識したうえで、既存ベンダーへのインタビューや既存のプログラム仕様書の確認などを、早い段階で実施することが望ましい	・法改正や制度変更等の頻度 ・対象業務の特殊性	・過去に発生した制度変更や規則改正等に伴う業務仕様の変更がデータに与える影響調査を上流工程の段階で確実にフォローする ・上流工程では、既存ベンダーを含む業務仕様に精通したメンバーの関与が不可欠である旨、受注の条件やプロジェクトの立上げ段階で明確に顧客側に伝える ・経営、営業部門を含めた課題管理の実施
55	中核と思われる機能以外に相応の開発規模が隠れていることも	他社パッケージ製品を採用したシステムのリプレースに際し、基本設計で入出力項目を調査したところ、パッケージを改造したり個別プログラムで補完するなど、パッケージ製品だけでは実現できない業務要件が存在することが判明した。しかしながら、商談の性格上、大幅なコスト増を顧客と折衝することもできず、相当部分を自社調達することになった	重点顧客に始めて自社製品を導入できる絶好の機会であったこともあり、業務要件よりもパッケージの機能比較に着目した提案を行ってしまった。短期開発であったこともあり、他社パッケージ製品に基づくスクリプトなどを解析する手間をかけるわけにもいかず、要件確認フェーズをクローズしてしまった	・メジャーなパッケージ製品に限って一般に公開されている情報量も多いこともあり、パッケージ製品の機能比較に偏った提案及び要件確認を行ってしまう傾向が強い ・パッケージ商談では、中核となる業務要件だけではなく、画面や帳票出力といった外部仕様への拘り度合いや、制度変更等に伴う処理の複雑さ等に関してもアンテナを高くしながら、上流工程全般をフォローするスタンスが重要となる	・要件定義の初期段階から、顧客側が細かい画面仕様への拘りを示すようになった ・対象業務の特殊性	・状況が許すのであれば、上流工程の早い段階でエンドユーザへのインタビューを行い、仕様変更の頻度や要求仕様への拘り度合いなどの把握に努める ・状況が許すのであれば、既存ベンダーへの事前ヒアリングやこれまでにかかったコスト（概要の工数でも可）の開示などを顧客側に依頼してみる ・既存システムのパッケージに関する技術仕様に詳しいメンバーをプロジェクトに投入する
56	ドキュメントの成果レベルは事前調整が不可欠	他社リプレース開発において、上流工程で実施する要件定義や基本設計などの成果イメージを顧客と合意していなかった。このため、設計過程で記載レベルや設計範囲などの調整に時間を要した結果、上流工程の終了が遅延し、コスト増加と後続の開発スケジュールに影響を与えた	初めて当該顧客の開発を請け負うこととなったため、ドキュメントの品質をはじめ、成果物の出来栄えまで事前にすり合わせができるような状況ではなかった。要件定義書や設計内容に関しては、記述内容の粒度や範囲について、どの顧客もあまり大きな差がないだろうという経験的判断が働き、成果物イメージのすり合わせを事前に行わなかった。要件そのものの追加や変更が発生することは想定していたが、要件定義書に記載する内容のレベル感が顧客と合っていないとは想定していなかった	・それぞれの顧客にとって、上流工程で作成するドキュメントの位置づけが様々である旨を理解することが重要である。顧客によっては、開発の標準化による手続きなどの都合で、他の顧客でいう詳細設計の一部を基本設計で求めるケースも少なくない ・特に新規顧客の場合は、顧客側で策定している開発プロセスに関する考え方や手続き等を把握することを最初のタスクとしてもよいほどである	・計画したとおりに打ち合わせが進行しない ・受注側では想定していなかったスケジュール遅延を顧客側から言い出し始めた	・プロジェクトのキックオフミーティングなどの機会を捉えて、顧客側が想定している開発プロセスや手続きに関するレビューを依頼する ・顧客側に既存システムのドキュメントについて開示を要求するか、受注側で成果物のイメージを共有できるサンプルの提示をもって、可能な限り、早期の意識合わせを行う ・経営、営業部門含めた課題管理の実施
57	RFPとの差異が明確になっているか	顧客から提示されたRFPに記載されていた機能の実現方法では業務要件を達成できないと判断し、RFPとの差異を明確にしないまま提案を行った結果、要件定義フェーズで顧客側との検討が噛み合わず、上流工程の完了が大幅に遅延したばかりか、改善提案部分のコストも持ち出す結果となってしまった	他社競合の商談であったこともあり、自社のソリューション力をPRすべく、一部の内容についてRFPが求める実現方式と異なる提案を行った。要件定義のベースが提案書記載の内容であるという受注側の思いを先行させたまま上流工程を進めたため、設計の後半まで、顧客との意識のずれを把握し切れなかった	・RFPの記載内容に対する改善提案をしたつもりでも、PowerPointによる概念的な表現が主流になっている昨今、そういた意図がストレートには伝わりにくい傾向がある。加えて、顧客側はRFPに準拠した提案である旨、潜在的な期待感もあり、相違点についてはより明確に記載するよう心掛ける必要がある。また、要件定義のベースがRFPなのか提案書なのか、といった基本感についても、顧客と早い段階で合意を得ておくことも大切 ・改善提案のなかには、システムの中核機能や顧客の現状に影響を与えかねない内容が含まれていることが多いので、プロジェクトの円滑な推進のため、早期にリスクとなり得る芽を摘んでおくことを心掛けておきたい	・仕様検討の過程で顧客の反応が鈍ってきた ・何度打ち合わせを重ねても顧客との合意を得られなくなってきた	RFPと異なる要件に関しては、顧客側との意識合わせを行う主旨からも、要件定義フェーズの前半に集中討議を実施するなどして、プロジェクトのリスクとなっていないことを確認する機会を設ける

事例番号	事例タイトル	概要	事例における見切りの内容	本来の見切りの考え方(※)	捉えるべき兆候	対処例
58	パッケージ開発では業務要件とのギャップが重要な見極めのポイント	パッケージ製品による開発に際し、プライマリーベンダを支援する形態でプロジェクトに参画したが、開発体制にパッケージ製品に関するエキスパートを組み込めないまま上流工程を進めたため、パッケージ機能と業務要件とのギャップが製造途上で判明し、方式全体の見直しを余儀なくされ、設計および製造に大幅な手戻りが発生した	＜プライマリーベンダーの見切り＞ 顧客のビル移転との関係で稼働時期が決められていたこともあり、パッケージ製品の適用による短期開発こそが最善の開発方式であると思い込んでいた。フィット＆ギャップ分析を意識はしていたものの、顧客側の仕様決定が度々滞る事態も発生したため、パッケージ機能を活用することで、顧客の要求仕様は概ね実現できるという直感的な見切りを行うことで、上流工程を完了させてしまった ＜サブベンダーの見切り＞ 採用しているパッケージ製品の製造元がプライマリーベンダーであったこともあり、プロジェクト推進過程で発生した問題解決は迅速に行えるはずという判断から、上流工程におけるエキスパート投入を重視しなかった	・パッケージ製品の機能に依存したシステム開発を行う場合は、製造工程を圧縮してでも、上流工程で業務要件とのフィット＆ギャップ分析を適切に実施すべきである。分析結果という明確な証跡をもって上流工程を完了することで、例え上流工程では見えない業務要件等が潜在していたとしても、中・下流工程で方式全体の見直しを行う場面で、コストや納期を含めた顧客やプライマリーベンダーとの折衝が容易になるものである ・自社、他社を問わず、パッケージ製品にかかわるエキスパートの投入や迅速なQ/A対応を実現できる体制作りは、パッケージ機能に依存したシステム開発の場合、避けて通ることができない重要な準備タスクである ・自社製品とは言え、企業の合併、吸収が頻繁なIT業界では、必ずしも製品のサポート体制が国内になるとは限らない。製品のサポートレベルについては、具体的な水準や実現時期を明確にしたうえで、プロジェクトへの参入時点で、プライマリーベンダーと体制に関する調整を済ませておくことが重要である	・機能設計やプログラム設計書の記載内容に曖昧さが残る ・パッケージ機能と業務要件とのマッピングに予想以上の時間を要し始めた	・業務要件とのギャップを埋めるノウハウを有する技術者を最優先で確保するようプライマリーベンダーに要請する ・上流工程で決定した設計内容の実現の可否を評価できるエキスパートを開発体制に組み入れることで、製造工程以降の手戻りを最小化するよう、プライマリーベンダーに要請する ・プライマリーベンダーとの契約(分担、責任等)を明確にしておく