ITプロジェクト

設計・製造・評価リスク

クイック・リファレンス

PMファクトリー　2016年

目次

はじめに

本書は、前著の「要件定義リスク　クイック・リファレンス」の後編として作成したもので、開発工程の中・後半である設計・製造および評価における主なリスクに焦点をあてたものです。

本書で取り上げた事例は、IPA／SEC（独立行政法人　情報処理推進機構　ソフトウェア・エンジニアリング・センター）が公開している失敗事例に対して、筆者にてそれらのリスクの回避策およびアクションのチェックポイントを示したものです。

本文中に記述されているリスク要因および番号は以下の通りです。

1） ヒトに関するリスク要因一覧

① 他者依存的姿勢（マネジメントリスク）
② 上位マネジメントの関与不足（マネジメントリスク）
③ ユーザーの参加・協力度不足（マネジメントリスク）
④ 組織能力不足（マネジメントリスク）
＊未熟な組織文化（情緒的人間関係、非科学的な情緒的思考、情報軽視、データ軽視、認識力の弱さ、コミュニケーション機能不全、丸投げ体質）
＊　戦略の欠如（問題に対する取り組みの姿勢、情報戦略）
⑤ 見積り能力（マネジメントリスク）
⑥ 要件定義能力（技術リスク）
⑦ リーダーのプロジェクトマネジメント能力（マネジメントリスク）
＊外部交渉能力
＊タイムマネジメント能力（時間の有効利用、アクションタイミングの適時性、ヒト・資金の資源投入時期・量のミスなど）
＊現場主義の励行（部下との常時コミュニケーションの実行）
＊見える化能力（見えない問題の顕在化能力）
⑧ メンバーの技術能力（技術リスク）、うっかりミスなどのヒューマンエラー（マネジメントリスク）
⑨ プロセス管理の有無（マネジメントリスク）
⑩ コミュニケーション能力（阻害・組織間ないしは人間間のギャップ）（マネジメントリスク）
⑪ 関連部署との連携不足（マネジメントリスク）

2） モノに関するリスク要因一覧

⑫ ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）
⑬ 開発のベースの有無（技術リスク）
⑭ 開発環境の不備（技術リスク）

3） 資金に関するリスク要因一覧

⑮ 開発費不足（マネジメントリスク）
⑯ 資源投入戦略の誤り・開発費投入時期ミス（マネジメントリスク）

4） 情報に関するリスク要因一覧

⑰ あいまいなスコープ。目的・開発範囲の不明確さ・複雑さ。
⑱ あいまいな要件。要件定義の不明確さ。
⑲ 情報の不備・不足

1. 設計の事前準備工程におけるリスク

事例１．　手抜きドキュメントによるオフショア開発

[事前準備リスク]

◎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

【事象概要】

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。

[SEC]

【判断の誤り】

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。[SEC]

【発生問題】 品質不良、進捗遅延

【リスク要因】

キーリスクは、①他者依存的姿勢（マネジメントリスク）、④組織能力不足（マネジメントリスク）および⑫ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）、にある。

関連リスクは下記の通りだ。

⑦ リーダーのプロジェクトマネジメント能力（マネジメントリスク）
⑩ コミュニケーション能力（阻害・組織間ないしは人間間のギャップ）（マネジメントリスク）
⑰ あいまいなスコープ。目的・開発範囲の不明確さ・複雑さ。
⑱ あいまいな要件。要求仕様の不明確さ。

【リスク発生源工程】 設計・製造の事前準備工程（オフショア開発）

【リスク回避策】

オフショア特有の問題ではない。何をつくるのかの定義もドキュメントもろくにそろってない状態で外注に出せば海外だろうが国内だろうが問題が起きるに決まっている。他者依存の丸投げの悪しき習慣の代表例と言える。

組織能力、特にその未熟な組織文化（非科学的な情緒的思考、情報軽視、データ軽視、認識力の弱さ、コミュニケーション機能不全、丸投げ体質）を改善しなければ国内はもとよりオフショアでも確実に問題を出し続けることになる。

【リスク回避のアクション・チェックリスト】

◎オフショア開発でのポイント

☐ ① ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の整備が必須。
☐ ② （オフショア）外注との正確なコミュニケーションを行うために正確なドキュメントを用いること。コミュニケーションの共通言語はドキュメントしかないという認識を持つこと。
☐ ③ 丸投げは必ず失敗するという認識をもつこと。
☐ ④ 元請ベンダーは製品の統合設計および統合検証を必ず実行し、最終的な品質責任を負うこと。

事例2.　他者依存の丸投げによるオフショア開発　　[事前準備リスク]

◎オフショアで品質と納期遅延の問題（発注側問題）
【事象概要】
　日本語が堪能なブリッジ SE がいるオフショア会社なので、従来の国内外注先と同じ仕様精度でプログラム設計～単体テストを発注した。忠実で才能あるプログラマは予定より早く納品したが、結合テスト以降、インターフェース・バグなどが多発。システム全体では納期を守れなかった。[SEC]
【判断の誤り】
「日本語が堪能」であることにより「日本の慣習・文化に熟知」と判断した。[SEC]

【発生問題】 品質不良、納期遅延

【リスク要因】
　キーリスクは、①他者依存的姿勢（マネジメントリスク）、④組織能力不足（マネジメントリスク）および⑫ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）、にある。
　関連リスクは下記の通りだ。
⑦ リーダーのプロジェクトマネジメント能力（マネジメントリスク）
⑩ コミュニケーション能力（阻害・組織間ないしは人間間のギャップ）（マネジメントリスク）
⑰ あいまいなスコープ。目的・開発範囲の不明確さ・複雑さ。
⑱ あいまいな要件。要求仕様の不明確さ。

【リスク発生源工程】 設計・製造の事前準備工程（オフショア開発）

【リスク回避策】
　事例1と基本的には同じ問題だと言える。判断の誤りの欄に記述されていることも間違っている。日本語が堪能で日本の慣習・文化に熟知していれば開発が問題なく実行されるなどと思うこと自体、自分の無知をさらけ出している。その論理が正しければ、日本語が堪能な日本人は全て優秀な開発者ということになってしまう。さらにここで言っている日本の慣習・文化の意味が分からない。ソフトウェア開発は日本の慣習・文化で実行されるべきものだろうか。そこで言っている日本の慣習・文化とは手抜きドキュメントのことを意味しているのだろうか。細かいことまで書かなくても行間を読めと言っているのだろうか。ソフトウェア開発と言うものは文化でするものではなくロジックという科学で実行すべきものだろう。これはソフトウェア開発に限らず組織的・科学的な仕事をする上での常識である。下請け会社に何もかも依存してしまった元請け企業の成れの果ての姿を見ているようだ。

【リスク回避のアクション・チェックリス】
◎オフショア開発でのポイント
☐ ① ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の整備は必須。
☐ ② オフショア外注との密なコミュニケーションを行うこと。コミュニケーションの共通言語はドキュメントしかないということ。
☐ ③ 丸投げは必ず失敗するという認識をもつこと。
☐④ 元請ベンダーは製品の統合設計および統合検証を必ず実行し、最終的な品質責任を負うこと。

事例3.　マネジメント・技術統制なしのオフショア開発　　[事前準備リスク]

◎オフショアで品質と納期遅延の問題（受注側問題？）
【事象概要】
　製造工程をオフショア会社に一括発注した。プログラマ個々人の才能はあり、予定より早く納品したが、コーディング規約など全体統制力に乏しく、後続工程で品質、納期遅延問題が発生した。[SEC]

【判断の誤り】
　プログラマ個々人の才能はあり、技術的には大きな問題はないと見切った。[SEC]

【発生問題】 品質不良、納期遅延

【リスク要因】
　キーリスクは、①他者依存的姿勢（マネジメントリスク）、④組織能力不足（マネジメントリスク）および⑫ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）、にある。

　関連リスクは下記の通りだ。
⑦ リーダーのプロジェクトマネジメント能力（マネジメントリスク）
⑩ コミュニケーション能力（阻害・組織間ないしは人間間のギャップ）（マネジメントリスク）
⑰ あいまいなスコープ。目的・開発範囲の不明確さ・複雑さ。
⑱ あいまいな要件。要求仕様の不明確さ。

【リスク発生源工程】 設計・製造の事前準備工程（オフショア開発）

【リスク回避策】
　品質不良や納期遅延の原因をオフショア先のコーディング規約の不備などの統制力不足をあげているようだが、それは違うような気がする。統制力というならば、発注元の組織なりプロマネにおけるオフショア先の統制力がないのではないかと思われる。発注元が持っているコーディング規約を何故出さないのか。また品質不良や納期遅延が、コーディング規約がないことだけで発生するとも思えない。おそらく要件定義書も不備で、プロセス管理もなされていなかったのだろう。この事例は受注側の問題ではなく発注側の問題だと言える。

【リスク回避のアクション・チェックリスト】

◎オフショア品質・納期問題対応
□ ① 元請け会社による統合プロジェクト管理の実施。
□ ② 明確なスコープの提示
□ ③ 明確な要件定義書の提供
□ ④ 基本的な開発ガイドラインの提供（開発プロセスガイドライン、設計手順書、コーディング規約、結合・総合テスト手順書など）
□ ⑤ 密なコミュニケーションの実施

2. 設計工程におけるリスク

事例４． 協力会社への丸投げによる結合の失敗

[設計リスク]

◎個別に検討したものを繋げてみたら、繋がらなかった

【事象概要】

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。[SEC]

【判断の誤り】

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。[SEC]

【発生問題】 損益悪化、納期遅延

【リスク要因】

キーリスクは、①他者依存的姿勢（マネジメントリスク）および④組織能力不足（マネジメントリスク）にある。

関連リスクは下記の通りだ。

⑦ リーダーのプロジェクトマネジメント能力（マネジメントリスク）
⑩ コミュニケーション能力（阻害・組織間ないしは人間間のギャップ）（マネジメントリスク）
⑪ 関連部署との連携不足（マネジメントリスク）
⑰ あいまいなスコープ。目的・開発範囲の不明確さ・複雑さ。
⑱ あいまいな要件。要求仕様の不明確さ。

【リスク発生源工程】 設計工程

【リスク回避策】

これは明らかにシステム設計ミスだ。いやシステム設計すら行われていなかった疑いすらある。元請のベンダーが複数の下請けベンダーに分割発注したようだが、元請においてプロジェクトとしての統合的な管理も技術的な指導もせず、さらにシステム全体の基幹設計もしないまま下請けに分割丸投げしただけではないのか。そうでなければ結合において深刻な整合性がとれないような状況にはならない。

次の文章が単なるジョークでないことがよく分かるだろう。

「【結合則の怪】 虫のあるものと虫のあるものを結合すると、虫のあるものになる。 虫のないものと虫のあるものを結合すると、虫のあるものになる。 虫のないものと虫のないものを結合すると、やはり虫のあるものになる」 [ソフトウェアの法則、木下恂著、中公新書]

【リスク回避のアクション・チェックリスト】

◎協力会社への丸投げ防止

□ ① 元請ベンダーはシステムの基幹設計およびシステム統合の責任を負う。この責任まで下請けに転嫁することはできない。
□ ② 丸投げは責任の放棄、仕事の放棄だという認識をもつこと。

事例5.　協力会社への丸投げによる設計・製造不良　[設計リスク]

◎規模が大きくなくても、協力会社に丸投げは危険

【事象概要】

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。[SEC]

【判断の誤り】

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。[SEC]

【発生問題】 品質不良、進捗遅延、損益悪化

【リスク要因】

キーリスクは、①他者依存的姿勢（マネジメントリスク）および④組織能力不足（マネジメントリスク）にある。

【リスク発生源工程】 設計工程

【リスク回避策】

規模が大きかろうが小さかろうが丸投げは必ず失敗を招くからダメなのだ。丸投げは仕事の放棄だという認識が全くないようだ。予算の関係上とか言ってはいるが目線がいい仕事することではなく利益獲得ばかりに向いているようだ。利益を追いかければ利益は逃げるという理屈が分かっていない。

準備段階においてちゃんとした社内開発体制を組まなかったことが失敗の最大要因だろう。規模にかかわらずそれ相応の社内体制および要件定義などのドキュメントはキッチリと準備する必要がある。自分の会社が中心になって仕事を成し遂げるという自律的姿勢や意思がまったく感じられない。こういった開発組織は情緒的思考、認識力の弱さしかなく結局は一定の利益を中抜きすれば後は外注に丸投げをすることが習慣化していくのだろう。

【リスク回避のアクション・チェックリスト】

◎協力会社への丸投げ防止

- □ ①　丸投げは仕事の放棄そのものだという認識を組織内で共有すること。
- □ ②　協力会社に対して作業は委託できるが製品責任は委託できないという認識を持つこと。
- □ ③　元請会社は外注に委託する仕事の内容に関して適切な理解と指導と検収の責任をもつこと。
- □ ④　プロジェクトの準備段階から社内に適切な開発および協力会社のコントロール体制を構築すること。
- □ ⑤　外注を有効に機能させたかったら要件定義書の精度を上げること。

事例6.　全面委託という名の協力会社への丸投げ　[設計リスク]

◎信頼関係だけでは適切な課題管理はできない

【事象概要】

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。[SEC]

【判断の誤り】

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。[SEC]

【発生問題】 設計工程大幅遅延

【リスク要因】

キーリスクは、①他者依存的姿勢（マネジメントリスク）、⑦リーダーのプロジェクトマネジメント能力（マネジメントリスク）、特にコミュニケーション機能不全、丸投げ体質、および⑫ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）にある。

関連リスクは下記の通りだ。

④ 組織能力不足（マネジメントリスク）
⑨ プロセス管理の有無（マネジメントリスク）
⑩ コミュニケーション能力（阻害・組織間ないしは人間間のギャップ）（マネジメントリスク）
⑰ あいまいなスコープ。目的・開発範囲の不明確さ・複雑さ。
⑱ あいまいな要件。要求仕様の不明確さ。

【リスク発生源工程】 設計工程

【リスク回避策】

この事例も丸投げ問題なのだ。基本設計以降を協力会社に全面委託した、と言っているが「全面委託」とは何を意味しているのでしょうか。責任もなにもかも委託したと言いたいのでしょうが、そんなことが許されるはずもありません。全面委託された協力会社がまた別の会社に詳細設計以降を全面委託したらどうなるのか考えてみるといい。この全面委託という言葉は典型的な他者依存の姿勢の表れと言えます。これは責任の放棄声明だと受け取れます。このようにして全面委託が繰り返されて、無責任体制の連鎖が完成するのです。これでまともな製品が生み出されるわけもないでしょう。この現象は多重請負構造という病理の一つなのです。

【リスク回避のアクション・チェックリスト】

◎協力会社への丸投げ防止

□ ① 丸投げは仕事の放棄そのものだという認識を組織内で共有すること。
□ ② 協力会社に対して作業は委託できるが製品責任は委託できないという認識を持つこと。
□ ③ 元請会社は外注に委託する仕事の内容に関して適切な理解と指導と検収の責任を持つこと。
□ ④ プロジェクトの準備段階から社内に適切な開発および協力会社のコントロール体制を構築すること。
□ ⑤ 外注を有効に機能させたかったら要件定義書の精度を上げること。

事例7.　あいまいな要件のまま協力会社へ丸投げ　　[設計リスク]

◎あいまいな仕様のまま実装し、テスト工程で要件不一致が多発
【事象概要】
　開発委託先からの納品を受けて結合テストを開始したが、要求機能を満たしていなかった。[SEC]
【問題点】
　要件（業務ルール）のドキュメント化が不十分で、プロジェクト・マネージャ（元請け会社）の頭にしかなかった。開発委託先へのレビュー不足。[SEC]

【発生問題】 品質不良
【リスク要因】
　キーリスクは、①他者依存的姿勢（マネジメントリスク）、⑦リーダーのプロジェクトマネジメント能力（マネジメントリスク）、特にコミュニケーション機能不全、丸投げ体質、および⑫ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）にある。

　関連リスクは下記の通りだ。
④　組織能力不足（マネジメントリスク）
⑨　プロセス管理の有無（マネジメントリスク）
⑩　コミュニケーション能力（阻害・組織間ないしは人間間のギャップ）（マネジメントリスク）
⑰　あいまいなスコープ。目的・開発範囲の不明確さ・複雑さ。
⑱　あいまいな要件。要求仕様の不明確さ。

【リスク発生源工程】 設計工程

【リスク回避策】
　この事例もまた丸投げ問題だ。あいまいな要求仕様書しか渡さず、技術的な指導もせず、必要なレビューもしないでまともに動くモノが納品されることを期待する方がおかしい。他人まかせの協力会社まかせの他者依存的丸投げの行き着くところはプロジェクトの失敗にとどまらず組織力の低下、技術力の低下による組織破壊にまで至ることでしょう。
　対策はオフショア開発でのポイントで指摘したことと同様で、下記のアクションを実行する必要があります。

【リスク回避のアクション・チェックリスト】
◎協力会社への丸投げ防止
☐　①　ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の整備は必須。
☐　②　協力会社との密なコミュニケーションを行うこと。コミュニケーションの共通言語はドキュメントしかないということ。
☐　③　丸投げは必ず失敗するという認識をもつこと。
☐　④　元請ベンダーは製品の統合設計および統合検証を必ず実行し、最終的な品質責任を負うこと。

事例8.　設計の中身に無関心なプロマネ　　[設計リスク]

◎設計の評価が終わっていない基本設計で協力会社に着手依頼
【事象概要】
　基本設計のフェーズが若干遅れてしまい、まだ設計の品質評価（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。
[SEC]
【判断の誤り】
　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。[SEC]

【発生問題】 損益悪化、設計品質不良

【リスク要因】
　キーリスクは、⑦リーダーのプロジェクトマネジメント能力不足（マネジメントリスク）にある。自分のチームの成果物内容の出来上がりレベルをよくみておらず、担当者に任せっぱなしのチーム内における丸投げと期限管理の甘さがプロジェクトを失敗させた。

　関連リスクは下記の通りだ。
④　組織能力不足（マネジメントリスク）
⑨　プロセス管理の有無（マネジメントリスク）

【リスク発生源工程】 設計工程

【リスク回避策】
　基本設計のレビューを行わないまま協力会社に詳細設計を依頼したが、基本設計の品質不良により大幅な手戻りが発生した。失敗の原因は開発リーダーが基本設計の仕上がり精度について何にも見ていなかったということだろう。フェーズ判定のためのレビューをやらなかっただけの話ではないでしょう。基本設計工程中に要件に対する整合性の検討会を一度もしなかったのだと思われる。

【リスク回避のアクション・チェックリスト】
◎設計の中身を見ていないプロマネ
□　①　プロマネは設計に限らず開発の各工程における成果物の中身の出来栄えについてキッチリと知っておくこと。これは開発の基本だ。
□　②　一発限りの形式的なレビューは何ら有効性をもたらさないということを知っておくこと。
□　③　プロマネは、工程ごとにその担当とコミュニケーションをすること。
□　④　公式期限の前に自分の自己期限を設定し、余裕をもって行動すること。

事例9.　危険行為に無頓着なプロマネ（フレームワークの改修）　　[設計リスク]

◎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

【事象概要】

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った。[SEC]

【判断の誤り】

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。[SEC]

【発生問題】 損益悪化、納期遅延

【リスク要因】

キーリスクは、⑦リーダーのプロジェクトマネジメント能力（マネジメントリスク）、特に見える化能力（見えない問題の顕在化能力）の不足にある。

関連リスクは下記の通りだ。

④ 組織能力不足（マネジメントリスク）

⑰ あいまいなスコープ。目的・開発範囲の不明確さ・複雑さ。

【リスク発生源工程】 設計工程

【リスク回避策】

基盤部分の修正は失敗すると往々にしてシステム全体に致命的な品質問題を引き起こすことは誰でも承知していることだろうと思う。

このベンダーは顧客要件が一部の基盤部の修正を伴ったため、ついでにメンテナンス性の向上もやってしまおうと考え基盤部に大幅な修正を加えてしまったのだろう。メンテナンス性の向上を図るためといっているからにはフレームワーク部分の構造に手を入れてしまったのだろう。このようなリスキーな改修は顧客の追加機能開発のついでにやるような安易なものではない。このようなリファクタリング開発は自社にて別途独立したプロジェクトにて実施し十分な検証をすませた後に顧客物件に適用するのが正道だろう。このプロマネは顧客開発を失敗の実験場にしてしまった。

【リスク回避のアクション・チェックリスト】

◎フレームワークの改修について

□ ① フレームワークの改修は顧客物件開発のついでに実行してはいけない。

□ ② フレームワークの改修は自社にて独立したプロジェクトにて実施し、十分な検証を経た後に顧客物件開発のインフラとして提供すること。

事例10.　フレームワーク構築の失敗　[設計リスク]

◎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
【事象概要】
　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。[SEC]

【判断の誤り】
　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。[SEC]

【発生問題】 損益悪化、納期遅延

【リスク要因】
　キーリスクは、⑧メンバーの技術能力（技術リスク）、および適切な人材の配置ができなかったリーダのプロマネ能力の欠如にある。

　関連リスクは下記の通りだ。
④　組織能力不足（マネジメントリスク）
⑦　リーダーのプロジェクトマネジメント能力（マネジメントリスク）　ヒト・資金の資源投入時期・質量のミスなど
⑲　情報の不備・不足

【リスク発生源工程】 設計工程

【リスク回避策】
　これはシステム設計の不良だ。この事例と同様の失敗をあちこちで大規模にやっている。ソフトウェアのメンテナンス性、カスタム開発の容易性を上げようとしてなんでもかんでも共通基盤やフレームワークにむりやりまとめてしまい、その結果適切な構造化に失敗し、業務APLとのインターフェースの整合性の設計に失敗し、異常にパフォーマンスおよびレスポンスが悪く性能不良で実用に耐えないものを作ってしまったのだろう。
　フレームワークの設計にはアプリケーションはもとよりミドルウェア・OSに精通した技術者の投入が必須である。常識を無視した先にあるものは非常識な現実だけである。

【リスク回避のアクション・チェックリスト】
◎フレームワーク開発失敗
□　①　フレームワークの設計にはアプリケーションはもとよりミドルウェア・OSに精通した技術者の投入を行うこと。

事例11．　異常系設計能力不足による失敗　　[設計リスク]

◎アプリケーション・プログラムの単体品質にばかりとらわれ、システム全体の納期・品質が守れず

【事象概要】

製造工程にて十分品質を確保したはずのアプリケーション・プログラム群が、総合テストでの多重アプリケーション・プログラム走行テスト中にアクセスの排他制御漏れによるデータ破壊などを多数引き起こした。土壇場で詳細設計からのやり直しが必要となり、システムの完成が大きく遅れた。[SEC]

【判断の誤り】

システム開発体制には経験豊富なアプリケーション・プログラム開発経験者が十分含まれており、OS やデータベースなどの環境は外部専門ベンダーに構築委託するので問題無いと見切っていた。[SEC]

【発生問題】 品質不良

【リスク要因】

キーリスクは、⑧メンバーの技術能力（技術リスク）および⑲情報の不備・不足（技術情報の貧困さ、失敗情報の蓄積不足）にある。

関連リスクは下記の通りだ。

⑫ ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）

【リスク発生源工程】 設計工程

【リスク回避策】

業務ロジックの出来栄えを表面的にいくら検証しても本当の品質の確保はできない。アプリケーションの表面には出てこない部分の設計および検証を怠れば不具合多発になるに決まっている。アプリケーションの背後に隠れているものには次のようなものがあるだろう。

排他処理不正、非同期制御不正、タイミングずれ、リトライ処理抜けまたは不足、タイマー値不正、レジストリー不正、設定間矛盾、Minimum／Maximum制御不正、メモリーリーク、など。

【リスク回避のアクション・チェックリスト】

◎アプリケーションの裏にあるもの

☐ 排他処理不正、☐ 非同期制御不正、☐ タイミングずれ、☐ リトライ処理抜けまたは不足

☐ タイマー値不正、☐ レジストリー不正、☐ 設定間矛盾

☐ メモリーリーク

【Mini／Max制御不正】

☐ メモリーリソース、☐ データ長、☐ 伝文長、☐ カウンター、☐ ポインターなど

もっと他にもあるかも知れない。全部チェックしなければアプリケーションはチャント動かないだろう。

事例12.　初歩的な設計考慮のミス（ログの無実装）　　[設計リスク]

◎担当者にシステム的認識欠落、障害時の切り分け、情報採取の仕組み不足
【事象概要】
　アプリケーション・プログラム担当者にシステム的認識がなく、性能、アプリケーション・プログラムに問題が発生した時の切り分け、情報採取などの配慮がされていないため、結合テスト、総合テストに手間取った。[SEC]

【問題点】
　システム系担当者が、アプリケーション・プログラムに手を加え対応した。[SEC]

【発生問題】 障害対応性の悪化

【リスク要因】
　キーリスクは、⑧メンバーの技術能力（技術リスク）および⑲情報の不備・不足（技術情報の貧困さ、失敗情報の蓄積不足）にある。

　関連リスクは下記の通りだ。
⑫ ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）

【リスク発生源工程】 設計工程

【リスク回避策】
　不具合検証や性能検証においては、手掛かりとなる各種のログが必須だということは平均的な技術者の常識である。ログがなければ不具合や性能の解析は不可能である。
　ソフトウェアにおけるログはソフトウェアの階層ごとに用意されていなければならない。要するにソフトウェア構造体のインターフェース部分には必ずログが必要となる。
　ソフトウェアの最下層ではハードウェアとのインターフェースログが必要となる。

【リスク回避のアクション・チェックリスト】

◎効率的検証には各種ログが必要
☐　キー操作ログ
☐　アプリケーションログ
☐　伝送通信ログ
☐　ミドルウェアログ
☐　I/Oドライバーログ

　必要に応じて、あるいは収集するデータの密度の粗さも含めて多彩なログが必要になってくる。

事例13.　オープンソフトウェア対応技術力不足　[設計リスク]

◎メーカー提供のソフトウェア部材の不具合でシステムが動かない
【事象概要】
　メーカー提供のソフトウェア部材を大規模に利用したが、メモリー・リークが多発して実用に耐えない。[SEC]

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

【発生問題】 システム動作不能

【リスク要因】
　キーリスクは、⑧メンバーの技術能力（技術リスク）および⑲情報の不備・不足（技術情報の貧困さ）にある。

【リスク発生源工程】 設計工程

【リスク回避策】
　汎用のオープン系OSやミドルウェアを使用する開発において採用したモジュールにおけるメモリー・リーク問題は基本的なチェック項目の一つであることは常識だろう。メモリー・リークに対する対策は次の通りだ。

【リスク回避のアクション・チェックリスト】
◎メモリー・リークの解消方法
□ ①　定期的・定時的に当該モジュールまたはプロセスをリブートすること。
□ ②　双子のモジュールを組込み、相互にスイッチ動作をさせリークを解消させること。
□ ③　メモリー・リークのない代替モジュールを探す、あるいは作成する。

　メモリー・リークはシステムのリブートあるいはプロセスのリブートにより解消される。

事例14.　パフォーマンス性能設計の不備　　[設計リスク]

◎大量データ処理方式の設計ミス
【事象概要】
　総合テストに向け、テスト環境とテスト計画を策定したが、アプリケーションの開発リーダーから性能問題に関する懸念があがった。[SEC]

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

【発生問題】 大量データ処理性能不良

【リスク要因】
　キーリスクは、⑧メンバーの技術能力（技術リスク）および⑲情報の不備・不足（技術情報の貧困さ）

　関連リスクは下記の通りだ。
⑫ ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）

【リスク発生源工程】 設計工程

【リスク回避策】
　この事例は、基本的には設計ミスによるものだ。システムとは”データ処理を行う仕組み”のことであり、データ処理とは次の二つの意味をもっている。
① 業務ロジックとして一次データを二次データに加工すること（ロジック要件）。
② 実運用に耐えられるように所定の時間内に処理を終了させること（性能要件）。
　さらに性能要件は、つきの二つで構成される。
① レスポンス性能
　レスポンス性能とは、操作あるいは処理に対する応答スピードのことであり、全体的な処理スピード（パフォーマンス）がいくら速くても操作に対する反応が遅いのは使いものにならない。
② パフォーマンス性能
　パフォーマンス性能とは全体の処理スピードのことであり、最大のデータ量を規定の時間内に処理する能力のことである。

　顧客におけるこれらの性能要件を事前に調査し、これらを満足させる設計・製造を行うことはシステムの基本要件中の基本である。

【リスク回避のアクション・チェックリスト】

◎システムの性能要件
□ ① レスポンス性能の達成。
□ ② パフォーマンス性能の達成。

事例15.　レスポンス・パフォーマンス性能設計の不備　[設計リスク]

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

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

【発生問題】 レスポンス・パフォーマンス性能不良

【リスク要因】
　キーリスクは、④組織能力不足（マネジメントリスク）、特に未熟な組織文化（非科学的な情緒的思考、情報軽視、データ軽視、認識力の弱さ）、⑦リーダーのプロジェクトマネジメント能力（マネジメントリスク）、現場主義の励行（部下との常時コミュニケーションの実行）および⑧メンバーの技術能力（技術リスク）にある。
　関連リスクは下記の通りだ。
⑲　情報の不備・不足

【リスク発生源工程】 設計工程

【リスク回避策】
　原因は技術者たちの認識不足による設計ミスによる一面もあるがプロマネは開発者たちの仕事内容を見ていなかったのだろうか。設計段階で設計者たちとのコミュニケーションあるいはレビューを行っていれば総合テストの段階で性能問題が発生することもなかっただろう。設計者の能力も問題だがプロジェクトを統制する役目のプロマネの能力がもっと問題だと言える。

【リスク回避のアクション・チェックリスト】
◎総合テスト工程での性能問題の発覚
□　①　性能設計ノウハウの習得および蓄積を行うこと。
□　②　全員による性能要件の重要性の再認識を行うこと。
□　③　プロマネと開発メンバー間のコニュニケーションの実施。
□　④　プロマネの自律性の育成。
□　⑤　プロセス管理の実施。プロマネによる設計レビューの実施。

3. 製造工程におけるリスク

事例１６．　危険行為に無頓着な開発者（ついで直しの失敗）
[製造リスク]

◎システムの品質が悪い
【事象概要】
　結合テスト時、単体テストレベルのバグが多数発生。結合テストを中断・延期し、工程遅延を招いた。[SEC]

【問題点】
　追加機能を開発していた際、メンテナンス性の向上を図るため、既存プログラムの修正も実施。その修正部分のテストを十分実施しないまま結合テストを実施した。[SEC]

【発生問題】 品質不良

【リスク要因】
　キーリスクは、⑦リーダーのプロジェクトマネジメント能力（マネジメントリスク）、特に見える化能力（見えない問題の顕在化能力）の不足なのだ。

　関連リスクは下記の通りだ。
④　組織能力不足（マネジメントリスク）
⑰　あいまいなスコープ。目的・開発範囲の不明確さ・複雑さ。

【リスク発生源工程】 製造工程

【リスク回避策】
　前述のフレームワーク改修の事例と同様の問題だ。たいした修正ではないとの甘い判断で、ついでにメンテナンス性もあげようと修正の影響度調査も不十分なまま既存ソフト部分に手をいれてしまったことによる失敗だ。ついでにという考え方で安易にソースコードをいじることは禁物なのだ。

【リスク回避のアクション・チェックリスト】
◎ついで直しは失敗の元となる
□　①　基本的に開発対象外のソースコードに手をいれてはいけない。
□　②　メンテナンス性の向上は別途腰をすえて十分に影響度を調査した上で実施すること。

事例17．　ルールなしのコーディング作業　　[製造リスク]

◎コーデ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
【事象概要】
　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。[SEC]

【判断の誤り】
　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。[SEC]

【発生問題】 レスポンス性能不良

【リスク要因】
　キーリスクは、⑦リーダーのプロジェクトマネジメント能力（マネジメントリスク）、特に見える化能力（見えない問題の顕在化能力）にある。

　関連リスクは下記の通りだ。
④　組織能力不足（マネジメントリスク）
⑧　メンバーの技術能力（技術リスク）、ヒューマンエラー（うっかりミスなど）（マネジメントリスク）
⑫　ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）
⑲　情報の不備・不足

【リスク発生源工程】 製造工程

【リスク回避策】
　コーディング規約は一つのプロジェクトだけの問題ではないだろう。複数のプロジェクトが常時実行されている開発組織では標準的なコーディング規約は必ず整備されていなければならない。
　この事例の問題は単にコーディング規約だけの問題ではない。適正なレスポンス・パフォーマンスを得るための設計のノウハウが欠如している。これらのノウハウは設計手順書などのガイドラインに集約されていなければならない。
　設計手順書、コーディング規約、結合テスト手順書、総合テスト手順書などは設計・製造・評価などにおけるノウハウ集でありチームにおける設計・製造・評価の思想および手法のガイドラインとなる。これらのガイドラインは設計・製造・評価における業務品質の均一化および業務の効率化を実現する。チームの技術者が各自勝手なやり方で設計・製造・評価したもので整合性をとることは殆んど不可能に近い。これらの手順書や規約などのガイドラインの整備は開発チームの必携のドキュメントと言える。もしこれらのガイドラインが無いようならば整備を急ぐ必要がある。

【リスク回避のアクション・チェックリスト】
◎レスポンス性能について
□　①　システム性能要件については設計手順書にノウハウを集約しておくこと。
□　②　満足すべきレスポンス・パフォーマンス性能は把握されているか。
□　③　レスポンス性能・パフォーマンス性能の実現にあたっては、顧客の実運用条件を熟知しておくこと。

事例18.　構成管理なしの混乱した開発プロセス　　[製造リスク]

◎結合テストで重複障害やデグレートが多発
【事象概要】
　結合テストに入り、バグを発見して修復しても、再度同じバグが発生したり、インタフェース・ミスが多発したりする。[SEC]

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

【発生問題】 品質不良

【リスク要因】
　キーリスクは、⑧メンバーの技術能力（技術リスク）および⑫ドキュメント（手順書など）の不備（技術リスク）
　関連リスクは下記の通りだ。
⑦ リーダーのプロジェクトマネジメント能力（マネジメントリスク）
⑫ ドキュメント（要件定義書・設計書・チェックリスト・手順書など）の不備（技術リスク）

【リスク発生源工程】 製造工程

【リスク回避策】
　要するに製造における構成管理などのルールや作法が何もない状態なのだろう。開発文化が貧困だとしか言えない。製造における無法状態の例を次に列挙してみる。
- ベースプログラム構造の理解不十分なまま、あるいは今回の変更対象の機能の理解不十分なまま、あるいはデータ処理の仕組みの理解不十分なまま、あるいは変更部分の他との関連影響度の理解不十分なままソースコード変更に着手している。
- 複数の開発者が関係しているのにも関わらず、早い段階で内容等の相互確認やレビューをせず、みなバラバラにソース修正にとりかかり、寄せ集めたソースで一気にテストをしている。
- 開発者同士お互いに誰がどこを変更したのか誰も知らない。
- 関数を理解不十分なまま無造作に使用している。
- 検討不十分なままソースコードの”コピー＆ペースト”を無節操に行なっている。
- 開発中に発見された、以前からの潜在バグについて検討・レビューもなく、いきなりソースコードの修正を行なっている。（清水吉男著、『「派生開発」を成功させるプロセス改善の技術と極意』）

【リスク回避のアクション・チェックリスト】
◎製造ルールについて
☐ ① いきなりソースコードに触れてはいけない。
☐ ② ソースコード修正手順書を作成すること。
☐ ③ 構成管理手順書を作成すること。
☐ ④ ソースコード修正手順書および構成管理手順書に従ったプログラミングを実施すること。

4. 評価工程におけるリスク

事例１９．　計画書も手順書もない評価業務
[評価リスク]

◎テスト作業に無駄が多い
【事象概要】
　作業手順の誤りやテスト・データの誤りによる再テスト、修復という無駄な作業が多発し、テスト用マシンの CPU 時間の半分以上を消費していた。これにより、テスト作業も遅延していた。[SEC]

【問題点】
・システム設定情報が誤っていた。
・テスト用データベース、テスト・データなどのテスト環境の品質不良。
・テスト手順や障害発生時の情報取得・解析手順、復旧手順、構成管理といったテスト作業の品質不良。[SEC]

【発生問題】 テスト業務品質不良

【リスク要因】
　キーリスクは、⑫ドキュメント（チェックリスト・手順書など）の不備（技術リスク）および⑦リーダーのプロジェクトマネジメント能力（マネジメントリスク）

　関連リスクは下記の通りだ。
⑧ メンバーの技術能力（技術リスク）、ヒューマンエラー（うっかりミスなど）（マネジメントリスク）
⑭ 開発環境の不備（技術リスク）

【リスク発生源工程】 評価工程

【リスク回避策】
　評価工程に入る前にテスト計画書やテスト手順書などの整備は必須である。プロマネや評価リーダはテスト開始にあたってテスト環境の準備やテスト計画について開発および評価メンバーと綿密な打ち合わせが必要である。

【リスク回避のアクション・チェックリスト】
◎計画性のないテスト業務
効率的かつ有効なテストを実施するためには下記が必要となる。
□ ① 適正なテスト計画書を用意しておくこと。
□ ② 適正なテスト手順書を用意しておくこと。
□ ③ 適正なテスト環境を用意しておくこと。
□ ④ 適正なテスト体制を確立しておくこと。
□ ⑤ 開発・評価両チームによる評価内容・日程などについての意識あわせを常時行うこと。

さいごに

事例全体を見渡して言えることは、それらの障害はさまざまな形で現れてはいるものの、それらの根本的な原因は、その当事者たちにおける実行すべき開発内容の把握不足と実行に必要なヒト・モノ・カネおよび時間の準備不足にあると言えます。またこれらの問題は多重請負構造やオフショア発注などによる工程間および組織間の責任意識を希薄にし、コミュニケーションの断絶により被害の程度をいっそう大きく拡大していると言えます。

これらの問題を端的に表している言葉が「委託」あるいは「一括発注」です。

これらの言葉をどのように理解しているかが勝敗の分かれ目となっています。お金を払って全部を任せたのだから任せられた方に全ての責任があると考えるのか、それとも任せたからといっても何をつくるのかという説明責任および製品として仕上げるための統合的プロジェクト管理は依然として自分たちにあると思うのかの違いです。失敗プロジェクトの大多数は前者の理解に流れた人々や組織によるものだと断言できます。

現実の世界を直視した場合、後者の考え方が最も合理的かつ妥当な考え方だと言えます。もしそうではないと思う方々は、今後も同じようなやり方を続けるとよいでしょう。その結果は三年後、五年後に抜き差しならない状態として自分の目の前に現れてくることでしょう。

参考文献・出典

＊紹介事例の「事例タイトル」、「事象概要」および「判断の誤り／問題点」についてはIPA／SEC発行『ITプロジェクトの「見える化」上流工程編（2007 年 4 月）・中流工程編（2008 年 8 月）・下流工程編（2006 年 5 月）』Copyright (c) 2010 IPA を引用。

＊[ソフトウェアの法則、木下恂著、中公新書]

＊清水吉男著、『「派生開発」を成功させるプロセス改善の技術と極意』、技術評論社刊

＊イラスト：いらすとや　http://www.irasutoya.com/