イマドキの『作らない開発』に潜む陥とし穴

JFPUG 技術担当役員
TIS株式会社 生産革新本部
品質マネジメント革新部
プロジェクトリスク監理室　井上 智史

最近よく聞く失敗事例

- 親会社のシステムをグループ企業に導入して業務を効率化しようとしたら業務が適合せず、カスタマイズで工数が膨れ上がった。
- 業界で評判のパッケージを導入したら、バグだらけだった。
  理由を確認すると、当社環境をサポートしている最新版は、
  開発されたばかりで他社で稼働実績のない初物だった。
- 全社の業務統一化のために、業務パッケージの導入を決めた。
  調査段階では当社業務との適合性は高いと評価されていたが、
  ユーザーテストを始めると足りない機能が次々と判明し、方針変更が必要になった。

問題の構造

他所で開発したものを流用する（作らない開発）

パッケージ適用、関連企業のシステム、オープンソース、SaaS ・・・

期待すること

- 自プロジェクトの開発工数が節約でき、開発効率が上がる
- 他所で成功実績のある業務アプリを使うことで、自社業務を高度化する

阻害要因

- 自社要件に基づいて作られた機能ではない
- 品質がブラックボックス

『作らない開発』の典型的な失敗ケース

～主にパッケージ適用を題材に～

ケース１：必要機能が不足する

- 例外的な取り扱いができない
- 自社の独自手順をサポートしていない
- 新しいバージョンでしか機能がサポートされていない
- 扱う業務の規模が異なる
- 海外パッケージの導入/国内アプリの海外への展開の場合
  - 前提とする法体系が異なる
  - 多言語対応されていない
  - サポートされていない通貨がある

⇒ 外部設計の途中で発覚し、仕様変更が拡大する

⇒ 発見が遅れると、ユーザーテストで見つかって、大きな手戻りになる

ケース２：業務を標準機能に合せられない

- 変化することへの抵抗
  - 現行仕様踏襲という圧力
  - 小さな操作性差異への不満
- 業務変革方針の不徹底
  - 変革の目的の不統一/変革意識の周知不足
- 業務有識者・意思決定者の不在
  - 業務の全体を把握し、機能の要/不要を判断できる有識者の欠如
  - プロジェクトを進めるための判断をできる責任者の欠如

⇒ ユーザー要求を無制限に取り入れ続け、開発工数が肥大化する

⇒ プロジェクトのゴールを決めることができず、何回もループする

ケース３：流用元の不適合/品質不良

- 環境差異による不適合
  - OS、DBMS、言語等、様々なレベルで製品間の相性があり、流用元との不適合が起こる ⇒ 適合させるための新規開発部分で品質不良を招きやすい
  - 環境の差異により性能が出ないケースもある
- 流用元に起因する品質問題
  - 元々、流用部分の品質が悪いケースがある
  - 流用先で「使わない機能」はテストが省略されやすく、不良が潜在する
  - 流用元でメンテナンス用の文書が整備されていない

イマドキの開発プロジェクトは何が違う？

- 現行業務が（ある程度）システム化されている
  ⇒ 新旧の業務移行が必ず発生する
  ⇒ 変わることへの抵抗意識が存在する
- 新システムを獲得するための方策が複数存在する
  ⇒ より効率的な方策への期待感が大きい
  ⇒ 成功事例やセールストークが強調され、リスクが見落とされる

※今回は触れませんが、他にも・・・

- アプリ開発以外にシステム基盤の更新/置換えが発生する
  ・・・・・・（以降は研究中）

トラブルプロジェクトに見られる問題行動と対応策

～ソフトウェアメトリクスに絡めて～

問題行動１：甘い見積り

手段A（パッケージ導入・システム横展開等）適用時の誤った見積りモデル

開発工数A ＝ 開発規模 ÷ 手段Aに期待する生産性

- 「作らない開発」による生産性向上効果は全工程に及ぶものではない
  ⇒「生産性 N割アップ」のような曖昧な設定は見積りとは呼べない
  ⇒工数削減効果を、工程毎に「どういう効果」が「どの範囲」に「どれだけ」影響するか、精緻に見積るべき
- リスクの抑え込み工数と、顕在化時の対応工数が見積られていない

正しい見積りモデル

手段A適用時の見積りモデル（期待値）

開発工数A ＝ 開発工数0 － 手段Aに期待する工数削減効果
＋ リスクA_1顕在化時の増加工数 × リスクA_1顕在化率
＋ リスクA_2顕在化時の増加工数 × リスクA_2顕在化率
＋ リスクA_3顕在化時の増加工数 × リスクA_3顕在化率
・・・・・

開発工数0 ＝ スクラッチ開発の場合の開発工数

現実には

開発工数A’ ＝ 開発工数０－手段Aに期待する工数削減効果
＋リスクの未然防止工数 ＋ リスク顕在化時の対策工数

さらに　対策工数の範囲を超える場合の代替計画を準備

問題行動２：不適切なFIT&GAP評価

FIT＆GAP分析　・・・　流用元（パッケージ等）機能と業務の適合率評価

「企画段階でFIT＆GAPが80%」

- 業務で実用に耐えることを確証するに足りる評価か？

⇒誰が何を見て評価した結果かを確認

（実用性の評価には、エンドユーザーの判断が必須）

⇒プロジェクトの初期段階で、充分な時間をかけて評価すべき

パッケージ適用プロジェクトにおける FIT&GAP分析の位置づけ

	標準機能の適用	ＦＩＴ＆ＧＡＰ分析	アドオン開発
企画・計画	・パッケージ選定 ・ベンダー調査	・予備評価	
要件定義	・パッケージを使った業務フロー定義	・適用可能範囲の確認 ・フィージビリティ検証	・パッケージ適用範囲外の要件定義
設計	・標準機能によるプロトタイピング ・実装方式設計 (実機検証を含む)	・業務とパッケージ機能の突合せによる精緻化	・アドオン機能の設計 ・パッケージ標準機能とアドオン機能のI/F設計
製作	・パラメータ設定 等		・アドオン機能の開発
テスト	・パッケージ標準機能とアドオン範囲の結合テスト ・システムテスト（以後、通常開発と同じ）		

IPA ソフトウェア高信頼化センター 高信頼性定量化部会 WG2(信頼性メトリクスWG)のパッケージ利用開発プロセス案を元に作成

問題行動３：「決められない」体制

失敗例：エンドユーザー部門による推進委員会

横並びの体制は・・・

- 全体を俯瞰する視点での意思決定者がいない
- 部門間の利害調整ができない
- ユーザー要望に歯止めがかからない

⇒ 最終決定者を明確にして、仕様決定/変更ルールを策定する。

⇒ 要望を出す役割と調整・抑制する役割を分ける。

⇒ 要求増大に論理的に歯止めをかける**閾値**を設定する。

FIT率による進行可否評価観点

工程	評価観点	閾値
要件定義 ～設計	想定した工数削減効果を得られるだけのFIT率を得られるか	企画・計画時点で想定したFIT率
製造～テスト	仕様変更によって、FIT率（≒工数削減効果）が損なわれていないか	ベースライン確定時に設定したFIT率

普段のプロジェクトマネジメントにおいて、閾値との差異をモニタリングし、超える見通しが見えた時点で、リカバリープロセスに入る。

意思決定のためのモニタリングに使える その他のソフトウェアメトリクス

- ベースライン規模からの乖離幅（増加率）
- 仕様変更数
- アドオン追加用予備費
- ＥＶＭ

イマドキの開発で失敗しないために

- イマドキの開発は従来と大きく異なる分けではないが、全く同じでもない。
  ⇒ 漫然と同じ進め方をするのではなく、開発プロセスを工夫する。
- 風評に惑わされずに、現実を冷静に見極める。
  ⇒ 効率の裏に隠れたリスクに備える。
- 適正に進められる条件を、定量的に見極める。
  ⇒ソフトウェアメトリクスを活用して、プロジェクトをコントロールする。