OLYMPUS

Your Vision, Our Future

短納期ソフトウェア開発を失敗させないための7つのヒント

2006年10月13日

アシストソリューション研究会　プロジェクト管理分科会
岩見　好博
（オリンパス株式会社　IT改革推進部）

Opto Digital Technology

問題提起

- ビジネスにスピードが求められ、Webベースシステムだけでなく、携帯電話やデジタルカメラ向け組込みソフトウェアも、2～3ヶ月で開発しないと競争に勝てない
- こうした短納期ソフトウェア開発プロジェクトではリカバリーの余裕がないため、初期計画の拙さ、進捗確認の遅れが即失敗につながる
- 長期プロジェクトよりも確実なプロジェクト管理が求められるとも考えられる
- では、XPやスクラムなどアジャイル開発におけるプロジェクト管理とはどういうものであろうか？

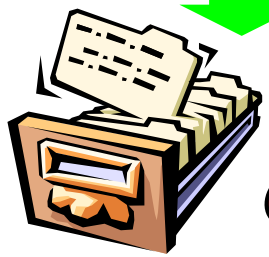
XPの作業プロセス

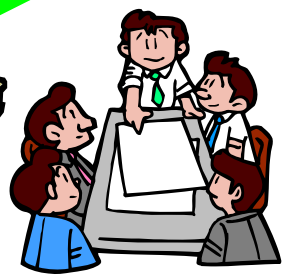
ユーザ

要件のネゴ

開発者

スコープ確認
最初の計画





ストーリー

全員設計

要件の共有

テスト

レビュー

ペアプログラミング

イテレーション1

イテレーション2

プロジェクト管理

イテレーションn

見積り、計画、追跡

見積り、計画、追跡

グラフの張り出し
スタンドアップミーティング

状況の共有

短納期プロジェクトとは？

- 短納期プロジェクトの定義
  - 開発期間3ヶ月、総工数30人月程度
    - 開発費用で3,000万円
    - コンカレント3～4チームで100人月も含める（開発費用1億円）
  - 工程範囲は
    - 要件定義からリリースまで6ヶ月
    - 詳細設計から結合テストまで3ヶ月（要件定義は別フェーズで行う）
    - 要件確認からリリースまで3ヶ月
- 短納期プロジェクトの属性
  - 社内システム開発か受託開発か
  - 経験の有無（業務、技術など）
  - 要件の確定度
  - 新規案件か、保守（改造）か

短納期≠小規模

短納期プロジェクトの問題点（反省）

- 要員（その分野に精通している技術者）のタイミングよい確保がむずかしい
- 複数社で開発する場合、ユーザの要件確認とプロジェクトのコントロールが難しい
- ユーザが要件を確定できない。あるいは、ユーザ自身が全業務フローを把握できていない
- 追加要件は取り込むべきでなかった。要件追加を認めたため、ただでさえ逼迫していた日程をさらに遅らせてしまった
- 初期要件をもっと確認しておくべきだった

表 5-2　短納期プロジェクト事例（補足済み）

プロジェクト名	プロジェクトの概要	開発期間	所要工数	内外作	社内受託	要件確定度	経験	新規改修	工程範囲	プロジェクト管理面で問題になった点		それをどう解決したか
										Risk	問題状況	（Riskを減らしていればこうなるために）
コミュニティシステム	Web系のコミュニティ開発	3ヶ月	60人月	外作	受託	中程度	業務なし技術あり	新規	詳細設計～総合テスト	リスク	（問題点） ・当社と他社開発のIF部分有開係が問題となる。 ・画面系開発（当社）とデータベース系開発（他社）の結合テストにおいて、レア仕様の取り決めに不整合が発覚した。例、今回、レア仕様にはXMLを使用した。 ・お客様要望により指定された当社と他社の開発の保存関係が字スト時、問題の引金となった。	（解決手段） ・他社のIFのキーマンに当社に来てもらい協同で問題の解決に当たった。問題のレア仕様について再度確認を行い修正し再テストを実施した。結合テスト時の開発体制、開発進捗の要素しを行うなどにより問題解決へと導いた。 （教訓） ・Web系開発の場合、開発分担の区切り方により問題の発生しやすい外部依存となる場合がある。そのため開発分担は独立したサブシステムで区切るのが望ましい。
営業支援パッケージ導入＆カスタマイズ	営業支援パッケージWebHeadを導入し、営業担当者のスケジュール、引合、顧客情報、訪問履歴などの日で一元管理することにより営業活動を支援する。（会社で実績のある） 日報、引合情報等の帳票プログラムを作成する。	4ヶ月	8人月	内作	社内	低い	業務ﾌﾛｰ技術○	改修	要件確認～リリース	タイム		パッケージの導入後、試行期間を2回に分け、試行1次はパッケージの実行環境を整えて、協力的な部署に試行してもらい、パッケージの機能を利用して、弊社環境を構築し、帳票系日報を追加。試行2次は他部署にも範囲を広げ試行しながら実行環境の整備を実施。
										人的資源	営業支援パッケージの導入だが、開発期間が短期であるにも関わらず営業関連知識を持つ開発担当者がおらず、環境設定、帳票開発に苦慮をきたした。	準備期間から1営業部門に参加してもらい問題解決を図った。ただし、開発期間2ヶ月では終わらず、本番移行後も調整が必要となった。（他の営業部署との調整、管理サイドからの要求）

問題点ごとの失敗要因と対処事例

問題点	失敗要因	対処事例
複数社で開発する場合、ユーザの要件確認とプロジェクトのコントロールが難しい	● 関係者が多いとコミュニケーションが難しくなり、必要な情報がタイムリーに伝わらなかった ● 下請けに丸投げされた	● 発注元、元請にコントローラを置いてもらう ● いない場合は、受注しないほうがよいかも
ユーザが、要件を確定できない あるいは、ユーザ自身が全業務フローを把握できていない	● ユーザに要件を確定する責任意識がない ● システム化に詳しいユーザは多くない	● ユーザ側で要件決定に責任を持つ人を明確にしておく ● 情報システム部門、または元請にユーザ要件を取り纏めるなど調整役となってもらう ● ユーザに要件を逆提案する

問題点ごとの失敗要因と対処事例

問題点	失敗要因	対処事例
要員（その分野に精通している技術者）のタイミングよい確保がむずかしい	● 短期調達だと単価が高くなり必要スキルを持つ要員を使えない ● 内作でも要員に余裕がなく、短期対応は難しい	● 内外作分析を行う ● 外作の場合は、一定の要員数を長期契約する ● ただし、コスト増、要員の引きとめ策などの課題が残る
要件追加を認めたため、ただでさえ逼迫していた日程をさらに遅らせてしまった	● 要件の変更管理が出来ていなかった（要件理解の差、要件追加か判然としない場合もある） ● イテレーション、スパイラル開発だったが、その定義が曖昧で混乱した	● 追加に伴う影響を見積もり、対応できるか判断する ● 要件変更による追加開発はせず、2 次開発に回す ● イテレーション、スパイラルの定義をユーザとはっきりさせておく
初期要件をもっと確認しておくべきだった	● 開発途中で要件や技術的前提について疑義が出た ● 双方の思い込みが発覚した	● 初期要件をユーザとよく確認し、文書化しておく

当たり前のことが出来ていなかった

失敗させないための7つのヒント

- 初期要件を絞り込む
  - 要件のフィルタリングが重要（どの要件のプライオリティが高いのか）
  - 顧客が要件の優先順位を決める期限を明確にしておく。
- 要件の確定度を高める
  - 要件の確定度を高めるフェーズを先行させる。
  - あらかじめ試行錯誤の必要性がある場合は、そのイテレーション計画をたてておく。
  - 顧客が要求仕様を作成し承認する期限を明確にしておく。
- 変更管理を確実に行う
  - 計画された期間内で対応できる場合に限って、仕様変更を受ける。
  - 期間内で対応できない場合は、納期変更や次開発に先送りする。

失敗させないための7つのヒント（2）

- リスク管理
  - リスクは無くしておく。残したままだと危うい。
    - » 例）不確実な新規基盤技術は、技術検証のプロタイプ実施計画を立てる。検証しても確信がもてないなら、その新技術は使わない。
- 要員確保
  - 必要な要員を必要な時期に確保するため、開発に必要なスキル、スキル別の要員数と時期をあらかじめ明確にしておく。
  - 必要なスキルの要員がアサインされているか確認する。
    - » 例）外注先に、開発担当者の氏名またはスキルを記載した開発計画書を見せてもらう。
  - 外注先と長期契約を結ぶ。
    - » プロジェクト単位の短期契約だと、人件費が割高で保守性も低下し、最悪の場合には必要な人員を確保できない可能性もある。
  - テスト結果の検証など内部レビューの工数と要員を予め確保しておく。

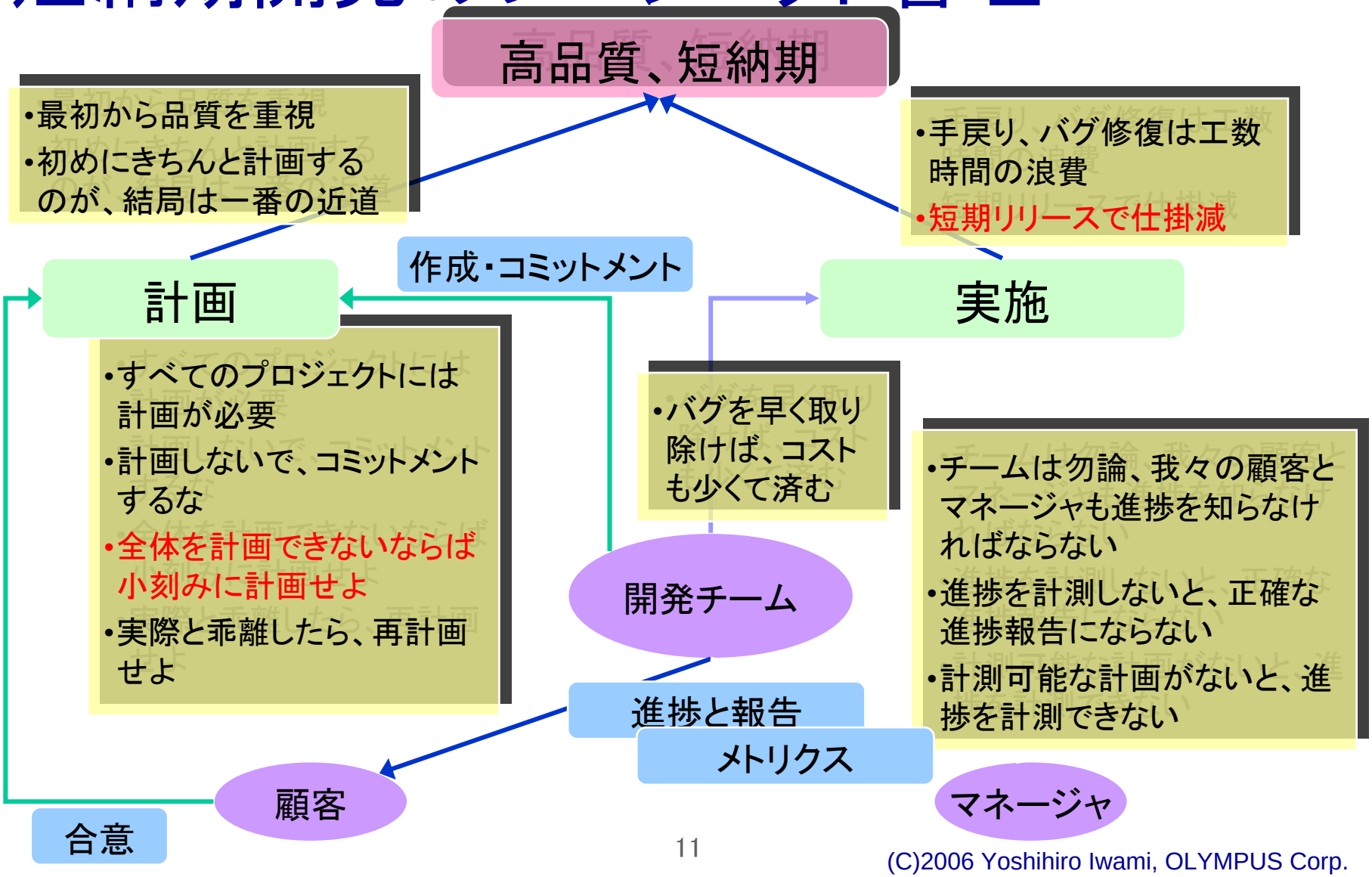
失敗させないための7つのヒント（3）

- 進捗管理
  - 外注の進捗をどのように把握するかを考えておく。
  - 開発状況の進捗報告は、何%終了ではなく、客観的な（何が終わっているか）報告とする。
    - » 計画と実績、および残存作業量を報告してもらう。
  - 検出バグ数を報告してもらう。
    - » 例）発注者向けにバグ情報をリアルタイムでWeb公開する
      米国では有料（契約金額の5%増）でも発注者は品質情報を求めている
- 外注先との関係
  - 外注先には、以下を選定すべきである。
    - » 当該テーマで実績を持つソフトウェア開発会社
    - » 長期に付き合いできる会社（後々、保守を依頼することを考慮して）
  - 元請からの丸投げを防ぐため「再委託」を認めない契約とする。
  - イテレーション計画のある場合、イテレーション毎の契約方式などを検討する。

7つのヒントを活かすには

- プロジェクト管理の基本を忠実に実行する
  - 例えば、短納期プロジェクトの計画作成でも、通常のプロジェクトと同様にマイルストーン、他プロジェクトとの依存関係を明確にする、など。
- 短納期であっても、個別開発を実施する前に「アーキテクチャー設計」あるいは「概念設計」フェーズが必要である
  - プロジェクト・メンバーが仕様や設計に共通理解を持てる。
- 短納期だからこそ、より適切なプロジェクト進捗管理を行うべき
  - 週次でなく、できれば日次で進捗管理を行う。メンバーの作業状況がリアルタイムでわかればベター。
- しかし、過度の文書化などを要求して重いプロセスにはしない
  - 必要最小限のドキュメンテーションレベルをユーザとあらかじめ合意しておく。

短納期開発のプロジェクト管理



まとめ

- 短納期プロジェクトだからプロジェクト管理は適当（やや手抜きして）にやっておけばよい、とは言えない
- 短納期だからこそプロジェクト管理がますます重要であることが確認できた。
- このヒントは、大規模開発にも適用できる
  - システム開発の全体像を明確にし、開発を独立した複数のサブシステムにわける。そして、サブシステム毎に開発優先順を明確にしてリリース時期をわける
  - インクリメンタル開発の考え方を導入して要件定義、外部設計を共通に行い、詳細設計～結合テストをコンカレントに行う

- 規模の大小や期間の長短に拘わらず、「よく計画する（planful）こと」、でないと「苦労する（painful）」を肝に銘じたい

ご清聴ありがとうございました