82

楽しいシステム開発、失敗しないシステム開発を実現する要件定義工程の進め方 [1]

概要

製造業大手企業から、「工場からの直販システムを短期間で開発したい。また運用時におけるシステム変更を迅速に行いたいので、保守についてはユーザー主体で行えるようなシステム開発の支援をお願いしたい。」という要望が入ってきた。エンドユーザーのシステム企画部門が主導して、要件定義工程からベンダー支援の下にシステム開発を推進した。

エンドユーザーと共同で要件定義を開始、翌週には利用者に操作が可能なプロトタイプシステムを提供し、その後エンドユーザーからの要望反映と確認を繰り返すアジャイル開発方式により、短期間でユーザーが納得できる業務の流れ、データの流れの確認ができた。要件定義終了時点で、ユーザーからこのまま運用で使えるのではないかと思われたレベルであった。利用者が見て触れ、修正要望が翌週にはシステムに反映されていることに驚き、喜び、進め方と進捗スピードに満足しているという声を聞き、SIer 担当も利用者の喜ぶ顔を見てやりがいがあふれた。

ツールを活用して開発方法を工夫し、従来の W/F 開発方法での課題問題を解決改善すべく目指した開発方法を紹介する。

1. 取組みの目的

よくシステム開発の失敗の話を聞くことがあり、システム開発業務は辛く厳しいというイメージがある。開発プロジェクトの失敗を回避するために、見えないリスクを考えて工数増加の見積り、管理のためのドキュメント作成（運用時には見ることがない）などが、システム開発コスト増大やシステム稼働までの長期スケジュール、開発技術者の長時間残業などを生み、企業経営へ悪影響を及ぼしている。その情報が学生にも伝わり新３Ｋとまでいわれ、IT 業界を敬遠する風潮も見え始めた。

システム開発及び運用保守において、従来の課題を下記に示す。

(1) 従来の W/F 開発方法では、テスト工程になってからユーザーが画面や操作性を確認する。そのためシステムイメージとの差異が発生することが頻繁で、その乖離を狭めるための修正追加作業は多大な工数コストとなり、時には上流工程に遡ってしまう、ユーザーと

[1] 事例提供：株式会社ウイング 樋山 証一 氏　周佐 匡芳氏

SIer の関係も悪くなり、お互いの経営に支障をきたすことも発生する。

(2) 稼働中の業務システムが運用環境（OS、DB などメーカーのサポート停止など）の変更により、システムを再構築せざるを得ないことが多く発生する。

(3) 利用部門が参加して、要件定義を行うことが難しい状況があった。また、情報システム部門が上流工程をまとめ上げる能力ノウハウがないことから、IT ベンダーへ上流工程から丸投げしている。

(4) システム開発の失敗を恐れ、リスクを過大に盛り込むことがあり、コストがかかる、スケジュールが長く納期が遅くなっている。

(5) 新しいシステム構築技術が次々に生まれ、習得が難しいことや将来の稼働を考え、環境や開発技術を採用する判断にも時間やコストをかけて検証することが必要。

(6) 上記により、開発技術者が残業過多や関係者との調整により、精神的に疲弊してしまうことが多い。

以上が今までのシステム開発での課題であり、解決すべき対象である。

これらの課題を上流工程において、ツールを活用することで、ユーザーの利用部門や情報システム担当が早期に業務やデータの流れ、操作性を確認できること及び変更要望に迅速に対応することで、要望をどんどん引き出しながら、まとめていくことを目指した。

また、ツールによりリポジトリ(設計情報)から、選択したプログラムソースコードやデータベースを自動生成することで開発工程のみならず保守工程にも迅速に対応できる。これは技術者の技術力習得や長時間業務の精神的な苦痛から救えるというメリットもある。

2. 取組みの対象、適用技術・手法、評価、計測

2.1. 取り組みの対象

工場の生産管理業務の前後工程の受注、部品発注、出荷業務の機能をシステム化対象として、要件定義工程を支援するものである。(図 82-1)

システムフロー全体図

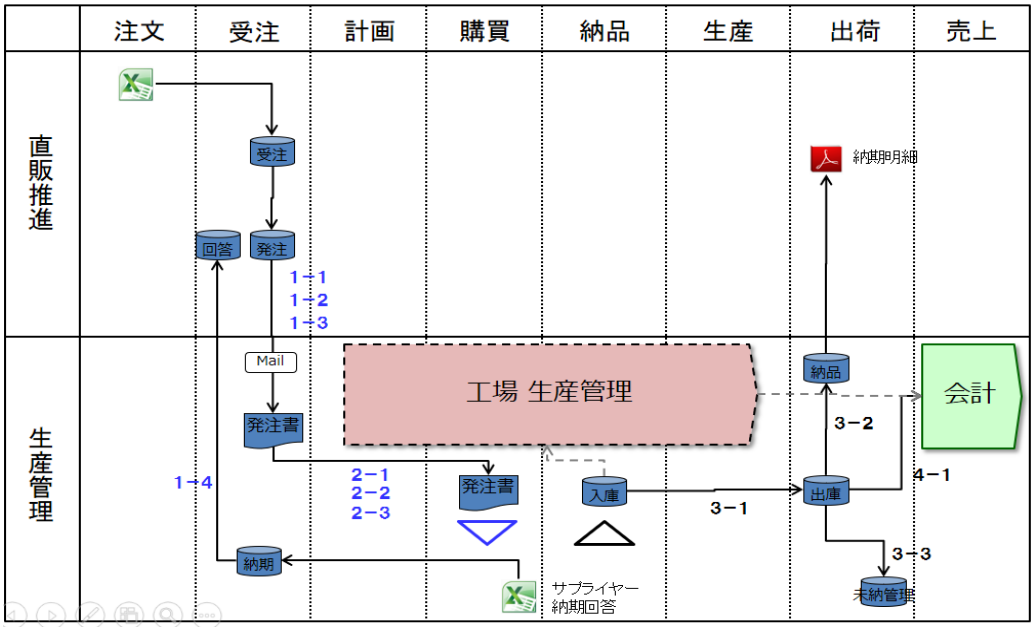


図 82-1 システム化対象範囲

2.2. 適用技術・手法

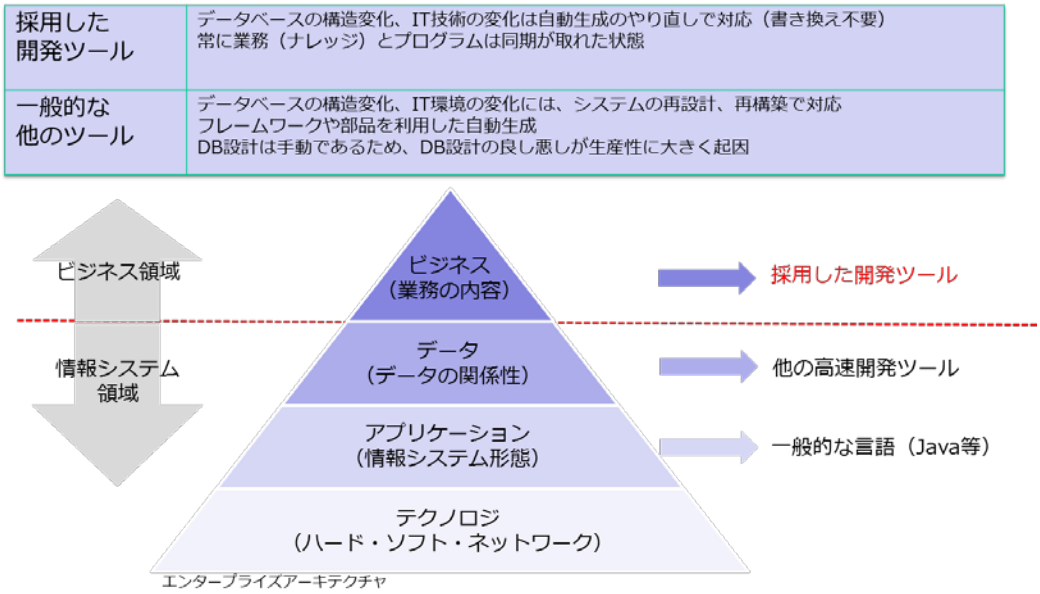
本システムの要件定義を進めるにあたり、システムを自動生成する開発ツール GeneXus[2]（以降「開発ツール」と表記）を採用した。他にもソースコードを生成するツールはあるが、生成したソースコードに技術者が手を加える必要があったり、DB 設計が手動でデータベース構造が変わった時に再構築の対応が生じたりすることがあり、IT 環境の変化、これから拡大するスマートデバイス対応もわずかな工数で対応できること、保守工程でも効果が大きいことからである。

開発ツール採用にあたっては、

(1) システムを 100%自動生成する開発ツールであること

“プログラムを書く”代わりに”現実の業務を記述”し、データベースと業務プログラムの自動的な設計、生成、保守を実現する。（図 82-2）

[2] GeneXus は、ウルグアイの IT ベンダーであるアルテッチ（Artech）が開発したアプリケーション自動生成ツールである。
URL https://www.genexus.com/products-japan/geneXus-jp?ja



※他のツールは情報システム領域(内部レベル)を設計したうえでシステムを自動生成するのに対し、採用した開発ツールは、ビジネス領域(外部レベル)情報を用いて、システム自動生成する。

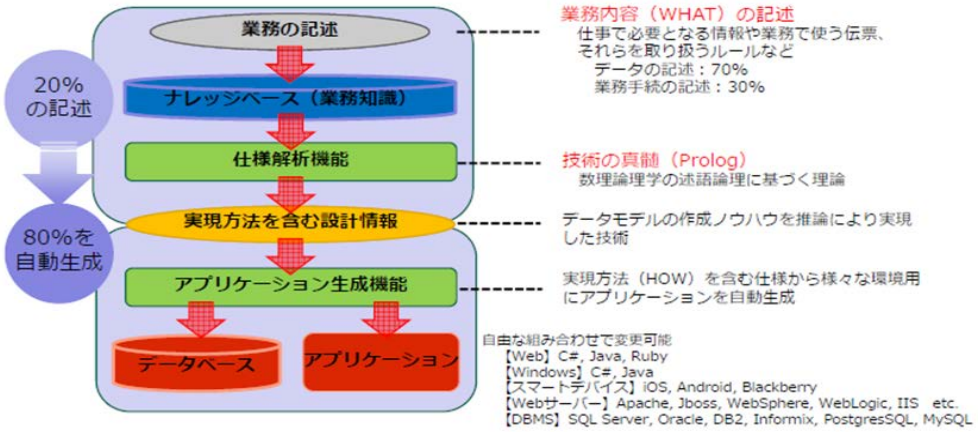
図 82-2 開発ツールの仕組み

(2) 業務視点のアプローチでプロトタイプ（試作品）を用いた開発であること

システム構築工程の早い段階でプロトタイプを用いて要件を確認でき、失敗しないシステム開発が早い段階で確信できるようになる。

(3) マルチプラットフォーム対応で稼働できること

ジェネレータの切り替えでマルチプラットフォームのシステムを実現でき、開発言語、DBMS の専門的知識は不要であり、最新の技術にもツールが対応してくれる。（図 82-3）



※投入された業務の記述を解析し、実現方法を含む設計情報を推論により自動生成する。生成された実現方法を含む設計情報に基づきアプリケーション生成機能が要求された実装環境用のアプリケーションと物理データベースを自動生成する。

図 82-3 開発ツールの位置付け

また、開発方法としては、プロトタイプを利用したアジャイル開発が最適であり、従来のドキュメントでの要件のレビューと合意、テスト段階で実物を見て検証ではなく、プロトタイプにより早い段階で業務フローに沿って繰り返しながら業務の確認と機能を実施して行く。最小限のドキュメントで認識祖語のないシステムを構築することを目指している。(図 82-4)

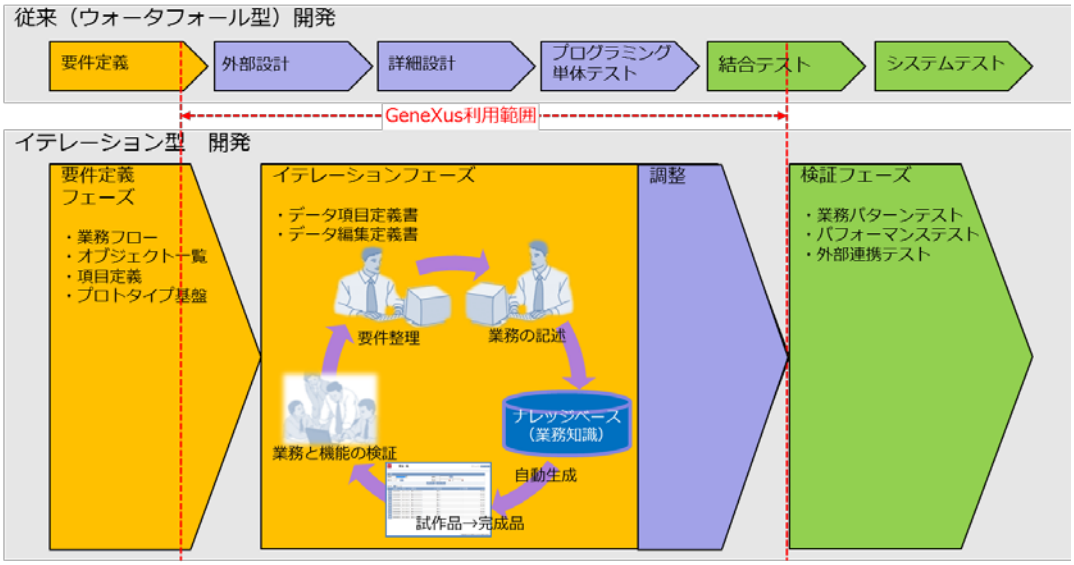


図 82-4 開発ツールを用いたアジャイル開発

2.3. 評価・計測

今回の取組みの評価指標は下記の通りである。

(1) 予定された期間、工数で要件定義を終了すること

(2) 利用部門が納得満足する操作性を実現すること

(3) ユーザー技術者がツールの活用技術を習得でき、保守工程で内製化できること

3. 取組みの実施、及び実施上の問題、対策・工夫

3.1. 計画・準備

プロジェクトの開発体制は、利用部門が要求・確認・テスト、システム企画グループがシステム設計業務、ベンダーが開発及び技術支援を行い、3 か月間でユーザーテストを完了するスケジュールであった。その内、初めの 2 か月間で要件定義及びバックロジック、帳票、他システムのインターフェースの実装を行った。　(図 82-5)

開発体制の要員に求められるスキルは次のとおりである。

実行環境に関する知識を持つ技術者（プロジェクト内に 1 名）

開発ツールの知識を持ち、開発ツールでの実現方法について提案できるシステムエンジニア（プロジェクト内に 1 名）

G 開発ツールでの開発ができるスキル(開発規模により複数名)

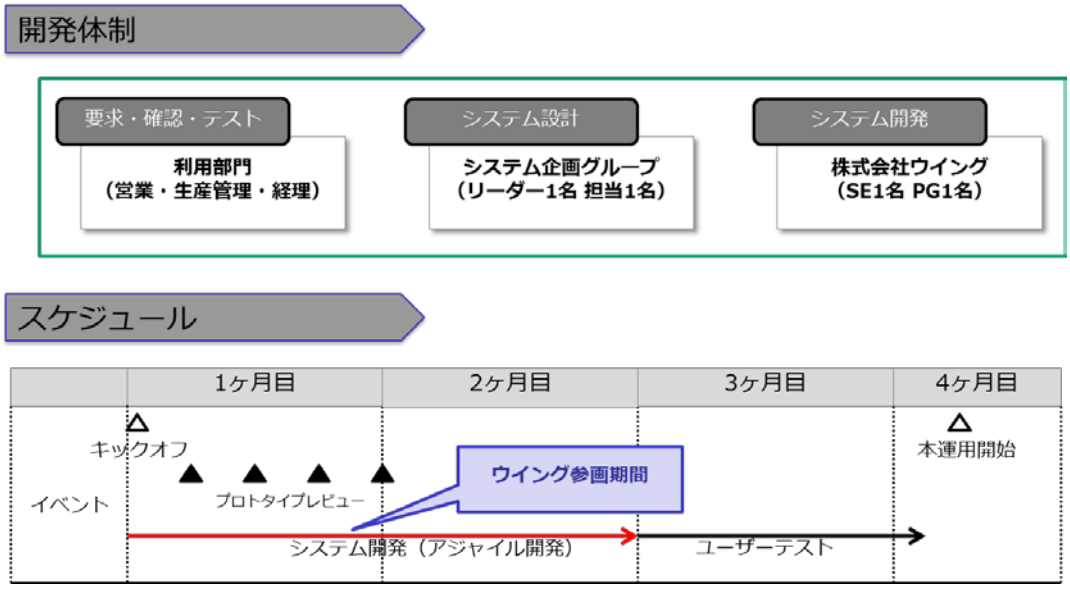


図 82-5 開発体制及びスケジュール

3.2. 具体的取組み内容

開発ツールに、業務情報を入力し自動生成された動くシステムをシステム企画者、利用者など立場が異なる部門にイテレーションの度ごとに確認いただき、その変更要望を迅速に動くシステムに反映させ、再度生成したものを翌週には見て触っていただくアジャイル開発の状況を示す。

3.2.1. 成果物

要件定義工程で作成した成果物は以下の通りである。

(1) 業務フロー

一般的な業務フローと同様に、登場人物と業務の流れ・タイミングを整理する。

最終的には各業務のシステム機能、その IN/OUT まで整理する。

(2) 機能一覧

上記業務フローをもとにシステム機能を洗い出した結果を一覧にする。

(3) データモデル

上記業務フローをもとに各業務の IN/OUT からデータモデルを作成する。

(4) データ項目定義

上記データモデルと帳票などをもとに、各データの項目を整理する。

(5) 機能概要

プロトタイプでは表現しない内容を記載する。

すべてのデータチェック、バックロジック、帳票の編集内容を記載する。

(6) 課題管理表

レビューで出てきた課題や要望を管理する。

(7) プロトタイプ

エンドユーザーとのイメージの共有のために作成する。開発工程ではこのプロトタイプをブラッシュアップすることで、システムを完成に近づける。

3.2.2. 1 週目の作業

システムの全体像をとらえ、プロトタイプを作成するために必要な最低限の情報を整理する。

(1) 業務フローの作成、確認

(2) 機能の洗い出し（機能一覧表に記載）

(3) 機能のデータ IN/OUT の洗い出し（データモデル作成）

(4) データ項目整理

(5) レビュー

ここまでの作業はシステム企画担当が作成し、利用者が回答者となってレビューを行った。利用者からは、「紙で見せられてもよくわからない」とのことであったが、この段階では詳細でなくともある程度の出来で進めてよい。

3.2.3. 2 週目の作業

開発ツールを使用してプロトタイプを作成開始。利用者にシステムのイメージを掴んでいただくために、

(1) 開発ツールシステム基盤を用意

(2) メニューデータ投入

(3) データ項目定義投入

(4) 各機能を、業務フローをふまえて調整

① 画面項目位置

② 検索条件、並び順

③ 必須項目

④ コンボ、ラジオ

次にプロトタイプが完成した段階で利用部門にシステムのイメージを見てもらい（プロトタイプレビュー）、次の作業を行った。

① プロトタイプを見ながら以下の項目を機能ごとにヒアリング

・ 項目過不足

・ 必須エラーチェック

・ 取得先データ

② 機能概要、課題管理表の作成

3.2.4. ３～５週目の作業

※２週目の作業内容を繰り返し

(1) 業務フロー

(2) 機能一覧

(3) データモデル

(4) データ項目定義

(5) 機能概要

(6) 課題管理表

(7) プロトタイプ

2 回目のプロトタイプでは「利用部門の誰にどの機能を使って業務を回していくか」を確認していただくとともに精度を上げていった。

利用部門でも、今まで俗人化していた行動のルール化や新しい要望も発生してきた。

また、全社のシステム規約があり、データの精度を保証するための確認機能や、ログ情報・セキュリティ等必要なものがあることが判明した。その機能を新たに盛り込むことになり、追加機能が膨らんだ。

3 回目のレビューになると、追加したい要件も増えてきたので、工数とスケジュールを踏まえながら将来的なものは将来へ、運用を回してみたうえで判断するように促していった。

重要機能は 2,3 回、簡易機能は 1,2 回繰り返すことで、機能が固まっていくようにレビューを進めた。

プロトタイプの活用はシステムのイメージを共有するためのものであり、齟齬が発生しないように、戻りが無いようにするためのものである。

そのために、イメージが見えないものは作らない。

例えば、

- バックロジックは目に見えない
- すべてのエラーチェックは不要
- 帳票も見た目だけで良い

そして、動作ができるＰＣを利用者に提供し、利用者に触ってもらうことを進める。

プロトタイプレビューの際、ユーザーから計 56 件の要望・課題が発生したが、52 件は期間内で対応を完了し、4 件は運用後に再検討することになった。

3.2.5. その後の機能追加、保守作業

採用した開発ツールは機能追加変更及び実行環境が変わるときに、ツールが影響分析してアプリやデータベースの再編成を自動で行うため、プロジェクト作業中にエンドユーザーシステム企画グループメンバーに開発ツールの基礎的な教育を行い、その後の機能追加、保守作業はほとんどメンバーが行うことが可能になった。

4. 達成度の評価、取組みの結果

4.1. 取組みの結果

取組みの結果として、図 82-6 のとおりの画面数、部品数を生成し、実開発工数を 3 人月で要件定義を終えた。当初は 23 機能であったが、プロタイプレビューを通して、38 機能に増えた。

機能数

分類	機能数（当初）	機能数（実績）
画面機能	19	32
帳票機能	2	4
バッチ機能	2	2
合計	23	38

実績値（詳細）

分類	数値	備考
業務画面数	32	
管理画面数	50	
共通画面数	28	弊社テンプレート利用
機能部品	142	内 8割が弊社テンプレート
要望件数	56	内 運用後に再検討 4件
開発工数	3人月	

図 82-6 システム開発規模

4.2. 取り組みの評価

機能数は当初の 1.5 倍以上に増えたが、スケジュールおよび工数は予定通りで要件定義工程を終了し、その期間の中でユーザーへの教育研修を行い、以降はユーザーだけで(ベンダーはアドバイス、質問の回答のみ)稼働までこぎつけた。

ユーザーからの評価は以下のとおりである。

- 利用者が一緒に作る感覚を持てるので、必要な機能を整理して検討できた
- システムがすぐに作れる（できている感覚）
- システムの画面、操作性に慣れているので、運用テストもスムーズに開始できた
- 要件定義で関係者の確認ができるので、大きなプロジェクトの失敗がない
- 要件定義後の開発工程のスケジュール、工数計画に大きなズレが発生しない
- 後工程が見える

ということが挙げられる。何よりユーザーと開発担当がシステム開発を通して、楽しくお互いの役割をこなし、良好な関係構築の中で共同作業を行えたことが最大の評価であり、効果であったかもしれない。

4.3. 開発の留意点

この開発においての留意点／反省点としては

- 要件定義工程がたいへん忙しい。次のレビュー（１週間後）がすぐに来る
- 利用者と開発者の打ち合わせが盛り上がる。全員が今のプロトタイプを如何に良くするかを考えるので、ボリュームが膨らむ傾向になる
- 工期、コストを考えながらまとめることが重要である(マネジメント能力)
- 要件をシステムに落としこむ能力があったほうがよい
- イテレーションを何回行うか、どのレベルまで行うか、当初から決めておくべきである

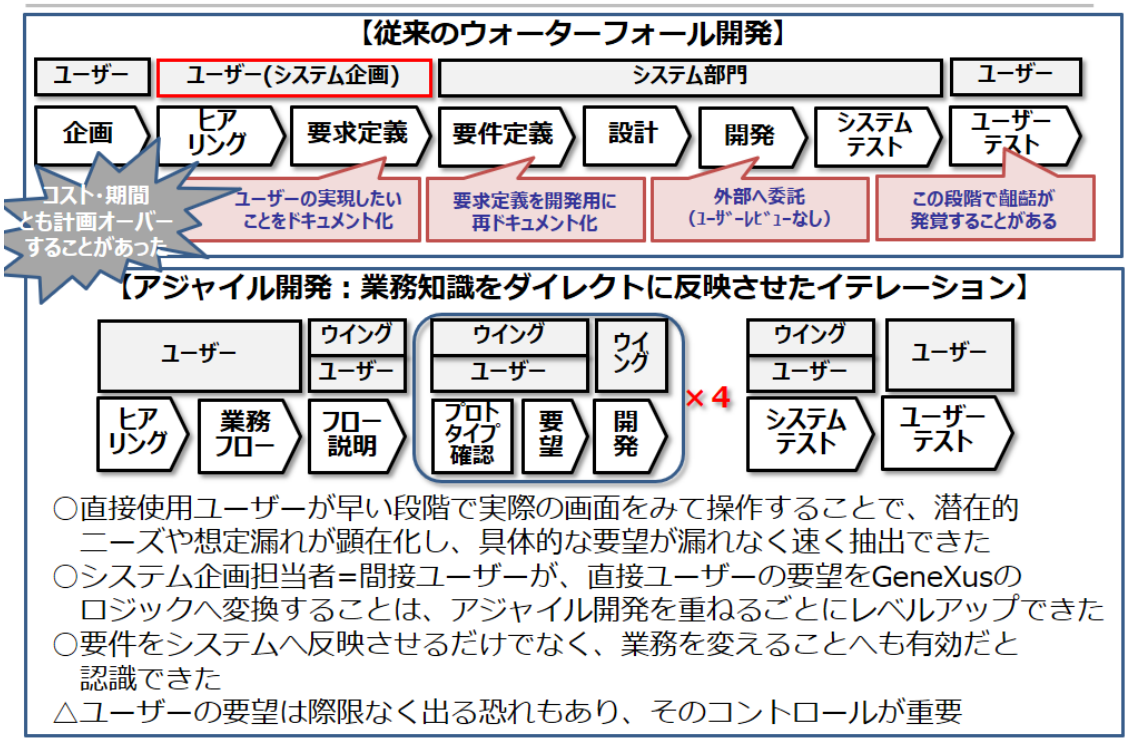


図 82-7 プロジェクトの進め方

4.3.1. ユーザーサイドのポイント

今回の開発を通じて、本開発手法が成功するためのポイントを示す。

(1) プロトタイプの目的は「業務の確認」であることを理解する

① プロトタイプは修正を繰り返すので、「現時点の品質」には拘らない

② プロトタイプは自動生成機能をフル活用する

③ 帳票・バッチ処理 等のイメージの齟齬が発生しないものは資料で確認する

(2) 開発ツールの自動生成・標準機能のメリットを理解する

① 自動生成機能・標準機能を活用することで、生産性の向上、品質の平準化が見込める

② リリース以降の保守メンテナンス性も向上する

(3) ユーザーと開発者で一緒に作り上げていく

① システム開発工程の中で、「要件定義／プロトタイプ検証」が最も重要な工程であるので、責任者（意思決定権者）を立て、途中で変更しない
② 責任者はシステム、仕様に対しての価値・コストメリットを判断する。コストメリットに合わない仕様・運用は実現しない（または優先度を下げる）

4.3.2. 開発サイドのポイント

(1) ユーザーの目線をもって、業務に対して最も良い実現方法を考える
① ユーザーの言うままに仕様を決めない
② 同業他社、過去事例などからユーザーへ提案する
③ レビューの場で提案を行う　（持ち帰って、検討して、資料を作って・・・では遅い）
(2) 常にトータルコストを把握しておく
① トータルコストがあればユーザーも優先順位や実現可否の判断が行いやすい
② 追加予算の判断、または機能増減による調整を随時行えるように準備しておく

5. 今後の取組みと考察(今後の課題)

5.1. 今後の取組み

本開発手法はその後、一年間の中で数社のユーザーの開発で適用し、ほぼ順調に開発を終えることができた。そして、ほとんどのユーザーから驚きと喜びをいただいている。この開発手法をIT 業界、とりわけユーザー企業に啓蒙していきたいが、拡大には課題もある。本開発手法ができる技術者の育成であり、工数での見積もりではなく価値に対する対価を得られるようにすることで IT 業界の変革につなげたい。

5.2. 今後の課題

早く開発ができるということは、ユーザーにとってはよいことであるが、従来の人月工数の契約を行うベンダーにとっては痛い話である。今後、この開発方法を普及していくには、以下のような取組みが必要である。

(1) 人月工数見積もりから、価値工数の見積もり方法に変更する、ユーザーから理解いただくこと
(2) 要件をシステムに落とし込む能力を持つ技術者の育成
(3) マネジメント能力を向上させること
(4) 本開発方法を啓蒙すること（既存の IT ベンダーの壁に阻まれることが多い）

5.3. 強まるユーザー内製化の傾向

今後、ユーザー企業の内製化、主体開発が増えると見込まれる。実際、弊社に依頼が来るユーザー内製化依頼はこの数年増えている。

ユーザーが内製化を進める理由は次の通りである。

- 事業を早く開始できる
- 適正コスト、IT ベンダーへの費用対価値が図られる
- 業務システムが IT ベンダーに握られるブラックボックス化を防ぐ
- 自社の情報システム担当が業務を理解するようになる
- 経営戦略の実現、IT 活用による新事業、ビジネスモデル創りが進む
- 情報システム担当者の価値と意欲と能力の向上が図られる

内製化を進めることで、ユーザー企業、情報システム部門は

- 経営戦略と IT が密接になる
- ユーザー(利用者）のシステム関心度が高くなり、より使いやすいシステムに改善していく
- 情報システム部門が経営と各現場(営業部門や製造部門)のハブになる
- 情報システム部門は頼りにされ、やりがいが増える、提案が生まれる
- 情報システム部門から役員になることが増える
- システム改善が早くなり、早く経営戦略に適応した経営活動が始められることで業績に直結する

情報システム部門は変化していくであろうし、IT ベンダーはそのユーザーの変化に対応して

- 下流工程(製造、テスト工程）の工数が減る
- 上流工程からの対応を求められる
- プログラミング技術よりシステム構築ノウハウを求められる
- ユーザー企業から直請が増え、多重下請け構造が変わる
- 開発よりサービス重視（教育、コンサルティング）
- 見積り方法が変わる(人月見積りから価値見積りへ)
- 学生、若手技術者の IT 業界の見方が変わる
- 人材育成のカリキュラムがプログラミング能力から上流工程、ユーザー対応能力の研修に移行する

という変化を起こし、技術者においても

- 上流工程対応業務が求められる（単なるプログラミング技術のみでは価値が低くなる）
- 受け身のプログラミング作業から主体的な動きでユーザー対応業務を行うことが求められる
- 成長が促進され、やりがいが増え、報酬も増える

に変わっていく期待がある。

ユーザー企業のスピード経営への支援を行うITベンダーの価値が増え、IT技術者は作業者ではなく、クリエイティブでやりがいの高い職種として再認識され、優秀な学生がIT業界を志望してくる。そんな時代が間近に迫っている夢をもって、本開発方法をお勧めしたい。

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（IPA/SEC）