93

中大規模エンタープライズシステム開発に適用可能なアジャイル開発手法 [1]

1. 概要

本編では、過去ウォータフォール開発での不採算案件における課題解決の施策として、ジェネレーションツールを用いたアジャイル開発を行っている、ＪＢＣＣ株式会社（以下、同社とする）の事例を紹介する。

2. 取組みの目的

2.1. ウォータフォール開発の課題

従来のウォータフォール開発では、要件定義、外部設計、内部設計、プログラミング、単体テスト、結合テスト、お客様テスト、パイロットテストと段階を経て開発をしていく為、お客様テスト段階になって初めて設計不備や障害（バグ）が発覚するケースが多かった（同社の場合）。この為、サービスインの大幅遅延や本番稼働後の大きなトラブル発生を過去に何度も繰り返してきた（図 93-1）。その大きな理由として次の５点が考えられる。

(1) 上流工程（要件定義～外部設計）でお客様の要件を聞き取れていない、またはお客様が要件を出しきれていない。

※お客様が要件を出し切れない理由（同社お客様の直接の言葉）

・ 現行仕様を把握しきれていない。

・ 本番データを使用しながら確認することができない。

(2) 紙（設計書）で確認することと、動くプログラムで確認することにおいて、場合によってはイメージが大幅に変わってくる。

(3) データ移行テストについては、お客様テストから初めて行われることが多い。

(4) 内部設計から結合テストまでの間、お客様側の関与が希薄となることで、お客様側から実態が正確に捉えられない（例えば進捗状況等）。

(5) お客様テストについてはスケジュールの一番最後となり、最も重要な指摘がこのタイミングで打ちあがってくる事となる。

また、開発全体期間を考えた場合、お客様テスト期間については一般的に短縮化されやすい。故に最終確認に十分な時間を確保できない現状がある。

---

[1] 事例提供：JBCC 株式会社 川上 明治 氏

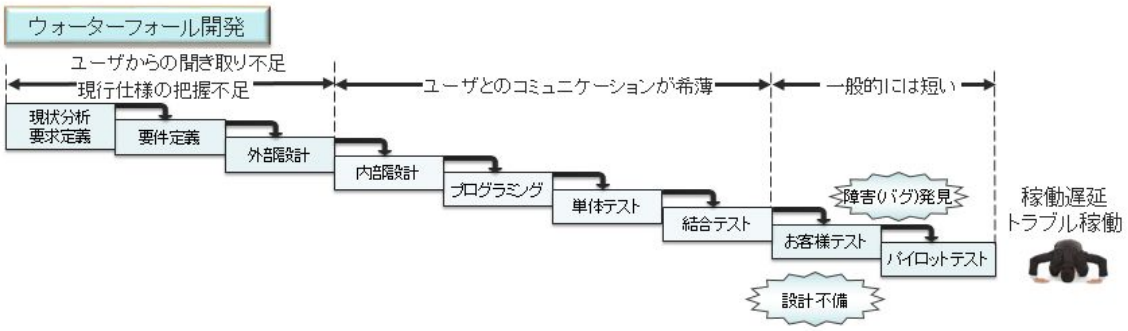


図 93-1 ウォーターフォール開発の課題

2.2. 同社の課題

これに加え同社が抱えていた課題として、次の 3 点が挙げられる。

(6) 開発期間の短縮化（お客様要望、同社要望の双方向から)。

(7) お客様より要望されるシステムが非常に複雑化してきており、結果、品質の確保が難しくなっている。

(8) お客様による自社開発（保守）への取組み要望への対処。

3. 取組みの対象、適用技術・手法、評価・計測

3.1. アジャイル開発への取組み

2014 年度以降に新規に開始する SI 案件について、基本は全て取組みの対象とした。アジャイル開発を採用する上で最も重要視したのはリスクヘッジの考え方である。昨今の非常に複雑化されがちなシステム化要望に対し、最後の最後に見せて大外しせぬよう、土台を作ったら見せ、肉付けしたらまた見せてといった手法を採ることで、従来の手法に比べ大幅な品質向上を期待した。

また、お客様からの要望として、内製強化・内製回帰があり、社内の要求としても若手技術者の早期戦力化（入社 1 年以内の現場投入）が求められた結果、開発ツールの採用を決めた。

アジャイル開発を行う上で、最も不安視したのがお客様とのアジャイル開発セッション中に発生するアベンド（Java で言う Exception）であった。これがアジャイル開発セッション途中で多々発生するようであれば、その都度アジャイル開発セッションが中断され、アジャイル開発がままならないと考えた為である。事前評価の結果、採用した開発ツールはこの問題を解決できるソリューションとして判断した。

この開発ツールは言語を知らなくても 1~2 ヶ月で習得できる為、新入社員であっても早期戦力となることが可能である。また、要件定義期間内で 2 ヶ月程度の開発ツール教育をお客様自身が受ける事により、プロトタイプ局面開始時からプロジェクトに参画し、開発していくこと

も可能と考えた。この為、新システムのサービスイン以降、お客様自身が保守していくことも可能と判断した。

なお、開発ツールの評価については、本ツールによる開発経験がある 3 社からの事前ヒアリング結果に伴った（アジャイル開発を進めていけるツールなのかというポイントで評価した)。

また、お客様からアジャイル開発を「支援契約」にすると、要望、障害が飽和せず、その状態のまま契約期間が終了するのではという不安点を挙げられた為、同社では「請負契約」で進めることとした。

4. 取組みの実施、及び実施上の問題、対策・工夫

4.1. JBCC アジャイル開発とは

一般的なアジャイル開発では、お客様と確認しながら作り上げていくことで、認識齟齬を早期に発見することが出来ることがメリットであるが、反面、次のような課題がある（図 93-2)。

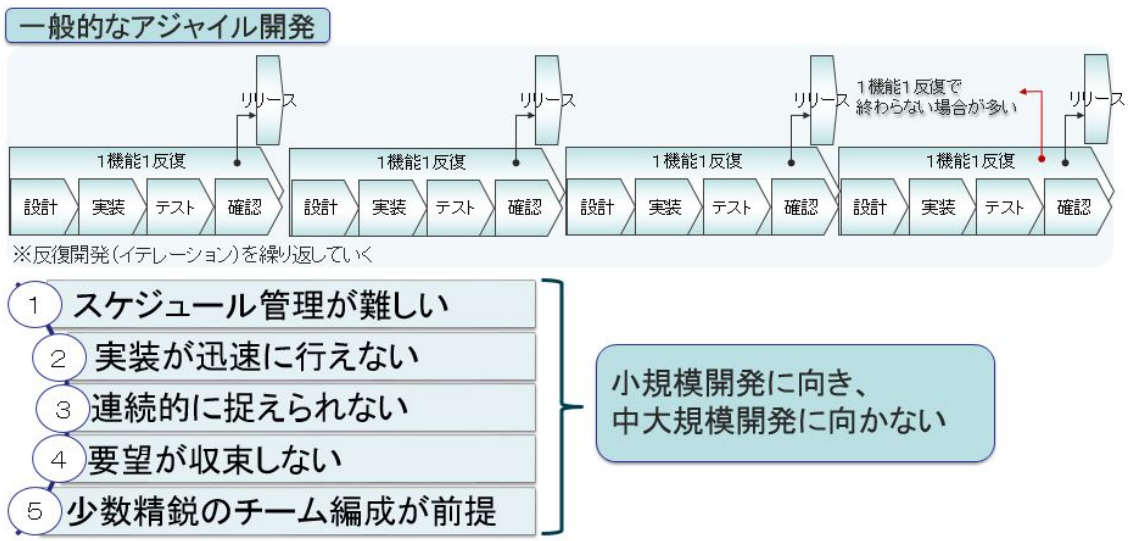


図 93-2 一般的なアジャイル開発の課題

そこで、同社では以下の工夫をすることにより、「JBCC アジャイル開発」として再定義する事とした（図 93-3)。

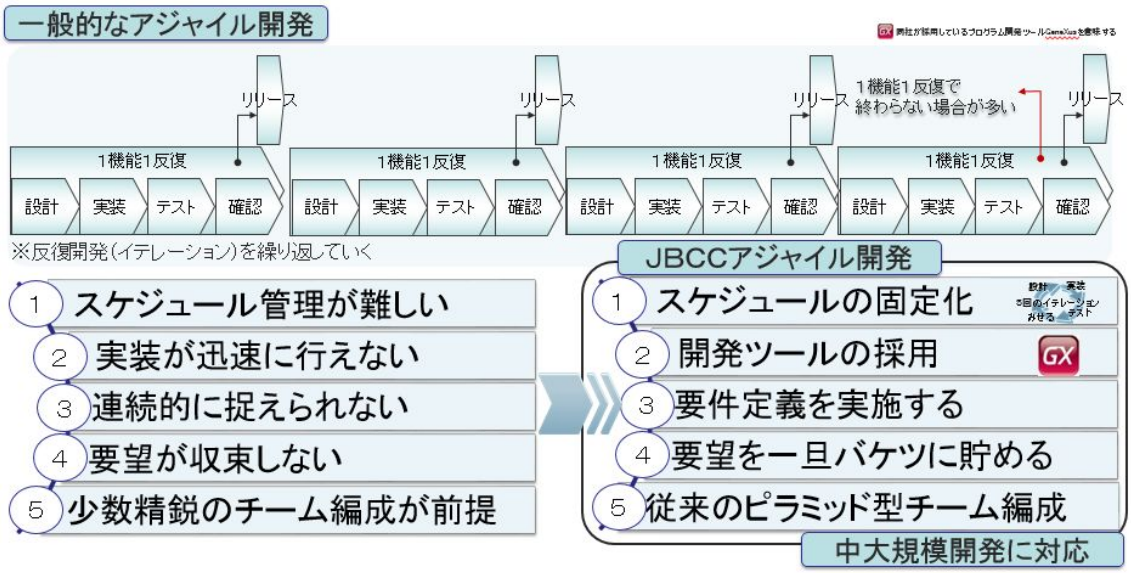


図 93-3 JBCC アジャイル開発の工夫

4.2. JBCC アジャイル開発の工夫

JBCC アジャイル開発の工夫についてもう少し掘り下げていく。

(1) スケジュールの固定化

現状分析、要件定義、プロトタイプ局面、プロダクト局面、パイロット局面で局面分割し、現状分析と要件定義は今まで通り（ウォータフォール開発と変化なし）、それ以降の開発局面において、「 設計⇒実装⇒テスト⇒見せる 」のイテレーションを、プロトタイプ局面で２回し、プロダクト局面で２回し、パイロット局面で１回しの計５回しで機能を構築していく事でスケジュールを固定化している（図 93-4）。

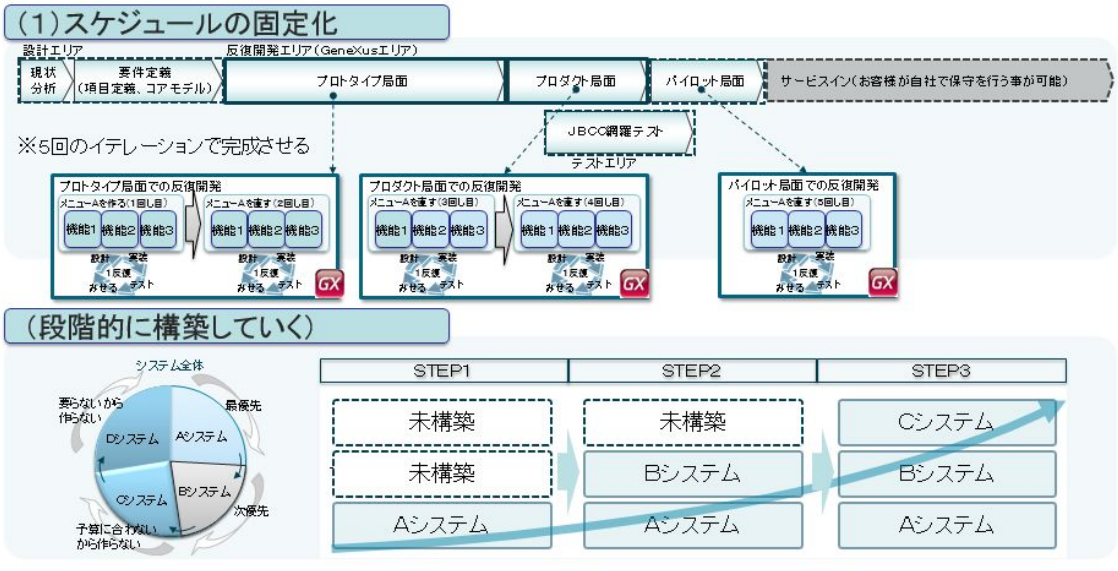


図 93-4 JBCC アジャイル開発の工夫（スケジュールの固定化）

各イテレーションにおいては、アジャイル開発セッション内で出されるお客様要望をA,B,C の３ランクに分け一旦は全量をストックする。その後、定期的なタイミングで対応の優先順位を判定しながら（取捨しながら）先に進めていく形をとっている。

※ A ランク：対応必須、障害（バグ）

B ランク：できれば対応したい

C ランク：後回しでよい

４年間での実践の結果、Ａランクはおよそ３~４回目のイテレーションで飽和に向かう。一方、B,C ランクは回を重ねる毎に上昇していく傾向がある（新システムに見慣れてくるので、それに応じていろいろな要望が増してくる)。ここがアジャイル開発の難しさの一つであるが、JBCC アジャイル開発では、プロジェクトのキャパシティーに応じて、A ランクは基本は完全対応、B,C ランクは可能な限り対応するという方法で B,C ランクへの対応を取り組んできた（図 93-5)。

結果、お客様のモチベーションとしては上々の状態をキープ出来ている。また、A ランクの対応がオーバーフローし、プロジェクトが頓挫するといった事態は、幸い今のところ経験していない。

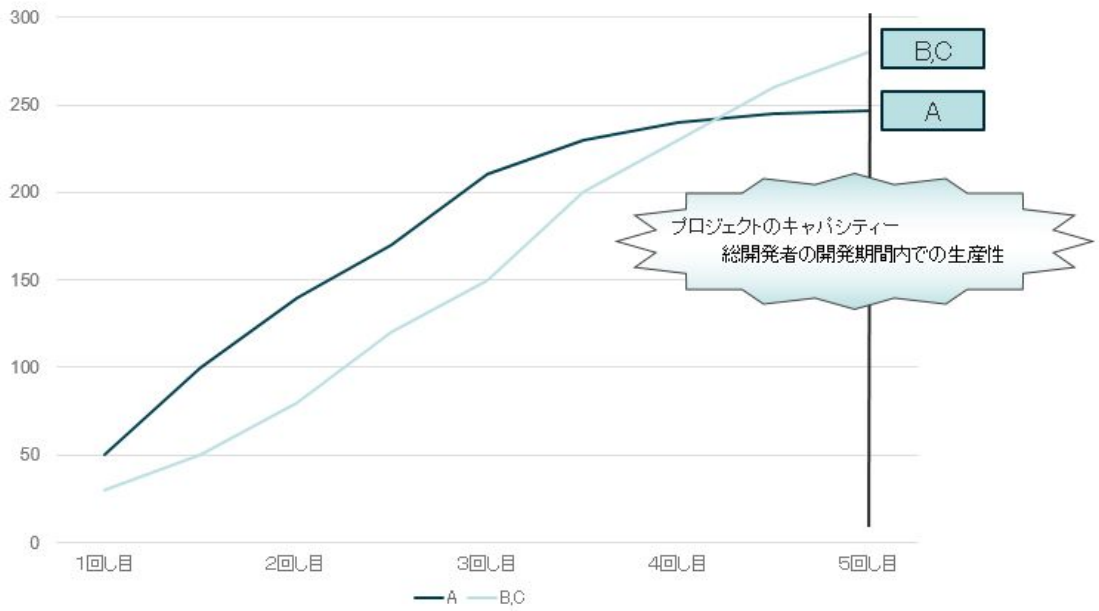


図 93-5 A ランク飽和曲線

(2)　開発ツールの採用

設計情報からプログラム（アプリケーション）を自動生成するツールである「GeneXus」を採用した。

物理バグの発生がほぼゼロになり、論理バグの検査に集中できるため、一定の品質を確保することができる（図 93-6)。物理バグとは、abend や exception エラーのことで、

本開発ツールでプログラムを作成する限り基本的には発生しない（なお、発生する条件については別途確認が必要)。よって起こりうるバグは、「日付入力域で 32 日以上を入力してもエラーにならない」「1 年が 356 日となっている」等の論理バグとなる。

(3)　要件定義の実施

一般的なアジャイル開発とは違い、現状分析や要件定義は従来通りに実施する。これにより新業務フロー（バッチであれば新処理フロー）までは確定させ（機能数までは確定させ)、開発局面以降のスコープ見極めを行っている。開発局面からは、この機能数を基に（ベースラインとして）JBCC アジャイル開発を行っていく（図 93-6)。

(4)　要望を一旦バケツに貯める

過去数年間の経験から、お客様の要望は尽きることがないという経験を得た。しかし、どこかでボーダーを定める必要がある。そこで同社としては、まずはお客様要望を抑えたり捨てたりせず一旦全てをバケツに貯め込み、全体把握の後に取捨や優先を判断し、その上で実装していくという方法をとる。

ここで言うバケツとは、プロジェクトのキャパシティー（総開発者の開発期間内での生産性）のことを指す（図 93-6)。

(5)　従来のピラミッド型チーム編成

明確に役割を分担することにより、プロジェクトマネージャーはプロジェクト管理に集中、セッションリーダーはお客様とのコミュニケーション（アジャイル開発セッション）に集中、設計者、実装者は設計、開発に集中させる。よって従来のウォータフォール開発時の体制と何ら変わりない体制で臨んでいる（図 93-6)。

結果、一般的なアジャイル開発のチーム編成に見られるような、設計できて実装できて話せてといったスーパーSE でチーム編成するような形を取らなくてすむよう対応できている。

なお、同社は RPG 技術者でウォータフォール開発を進めてきたメンバーが多数を占めていたので、当初は開発ツール教育（自社 OJT）を 2 ヶ月受けてスタートしていた。よって取り組み始めたばかりの時には、作成ランクが難しいものは外部委託によりまかなってきたが、現在では難しいランクを担当できるメンバーも育ち、同社のメンバーのみで構成することができるようになっている。

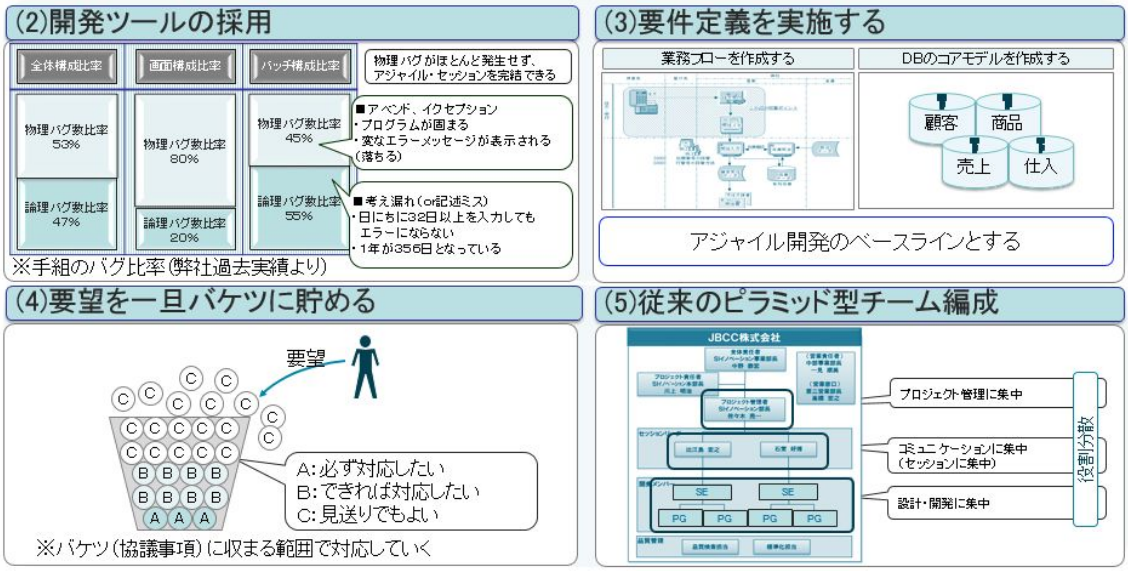
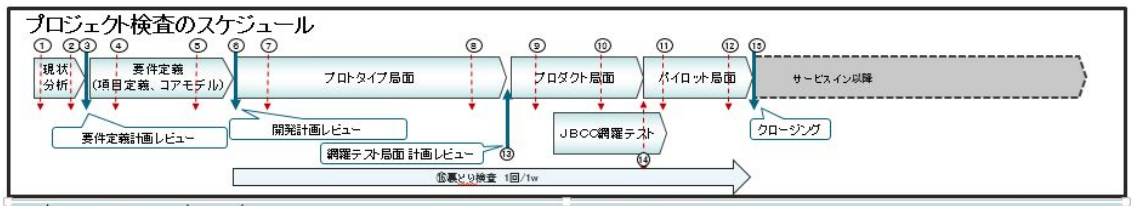


図 93-6 JBCC アジャイル開発の工夫（その他）

4.3. 問題

同社では、これらのプロジェクト運営において、品質管理の観点から定められた適切な時期にプロジェクト検査を実施してきた（図 93-7）。

基本的にはプロジェクト検査用のチェックシートを用いて、紳士協定（プロジェクトメンバーが虚偽報告（誤報告を含む）を行わない）の下、PM、メンバーに対し口頭確認で実施してきた。



No.	局面	時期	特に重点的に検査する内容
①	現状分析	前半	分析方法（ヒアリング有無、削除、お客様体制など）
②	現状分析	後半	全体業務F、機能削除、ビジネス計画
③	要件定義 計画レビュー	-	社内の品質検査機関（第三者機関）が行う品質検査
④	要件定義	前半	言葉の定義、業務の聞き取り度合い、分析方法（ヒアリング有無、削除、お客様体制など）
⑤	要件定義	後半	非機能、移行方針、環境構築、次局面WBS、全体業務F、コアモデル、システムの簡素化、平準化モデル、ウォークスルー評価シート（お客様、JBCC）、部品化
⑥	開発計画レビュー	-	社内の品質検査機関（第三者機関）が行う品質検査
⑦	プロトタイプ	前半	項目定義と実装の乖離、品質検査（画面は協議事項表、バッチはそのもの確認）
⑧	プロトタイプ	後半	データ移行、横展開、要望推移、検知障害、品質検査（画面は協議事項表、バッチはそのもの確認）、脆弱性
⑨	プロダクト	前半	障害分析
⑩	プロダクト	後半	クライテリア運用方針と細分化、パイロット計画（実施メンバ）
⑪	パイロット	前半	
⑫	パイロット	後半	クライテリア達成状況、LCC契約
⑬	テスト計画レビュー	-	テスト計画
⑭	網羅テスト	後半	障害分析
⑮	クロージング	-	標準化への寄与採点
⑯	裏どり検査	随時	疎通確認、実装確認、パフォーマンス対策、新旧対比、横展開

図 93-7 プロジェクト検査スケジュール

しかしながら、このようなプロジェクト検査を実施してきたにも関わらず、トラブルプロジ

ェクトとなってしまった案件がいくつかあった。

プロジェクトを進める中で非常に苦労した案件と、実際のトラブル案件を合わせた場合、２０１６年度のJBCCアジャイル開発案件では３１％を占めた（なお、実トラブル案件は６％である）（図93-8）。

苦労した案件と実際のトラブル案件を合わせた３１％の主原因として、以下5項目が挙げられる。

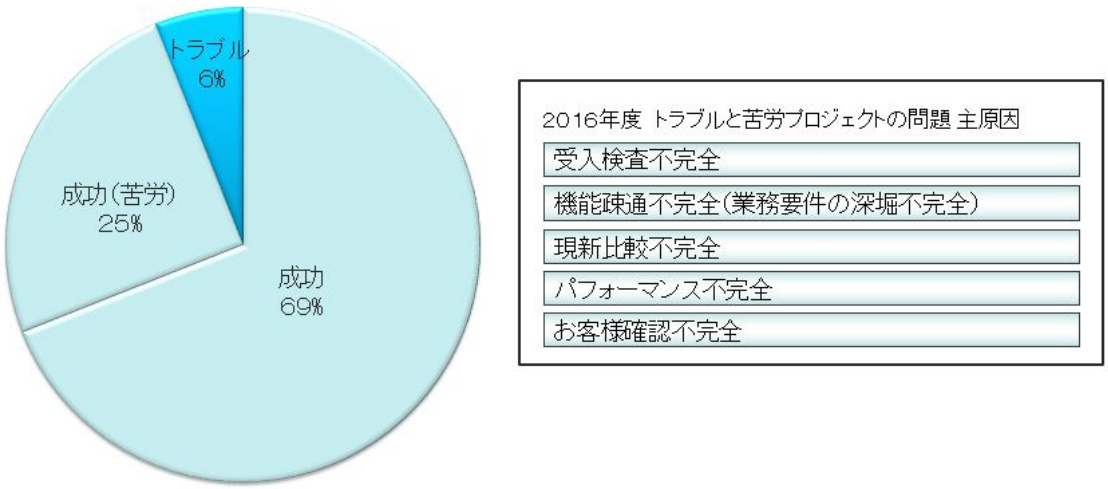


図93-8 ２０１６年度JBCCアジャイル開発プロジェクト状況

問題の主原因になっていた５つの要素の本来あるべき基本ルールを以下に示す。

（これらの基本ルールに則った形でプロジェクトが遂行されていなかったことがトラブルや苦労の原因であった）

(1) 受入検査

各プログラムを実装者が作成（修正）した後、アジャイル開発セッションを始める前、設計者もしくはPMが受入検査を実施する。受入検査の目的は、アジャイル開発セッションを全う出来るかどうかの判定を主としている。

(2) 機能疎通検査

JBCCアジャイル開発におけるアジャイル開発セッションの主目的は、機能の深堀である（機能性の追求）。この為、アジャイル開発セッション実施前の準備として最も重要視しているのは、機能疎通の担保である。

例えば、画面内容の確認において中途で障害（バグ）が発生しても問題検知として一旦留保し、先に進める形を取っているが、次の画面に遷移できないとか、出るべき帳票が出てこないといった場合、先に進めることが出来なくなり、結果、機能の深堀（機能性の追求）が完遂できないことになる。

この為、事前準備においては、作られたばかりのプログラムに対し障害（バグ）の優先判定を行い、機能疎通に関わるものを優先的に修正し、それ以外は一旦放置して（検知のみ）アジャイル開発セッションに臨むルールとしている（図 93-9）。

区分	協議事項内容			分類	補足・具体例	ＰＧ修正必須	対応優先度	設計者確認必須
障害	稼働確認できない	処理が異常終了などで、処理結果を全く確認できない。	①	バッチ	異常終了する。正常終了するがデータが登録(更新、削除）されない。 ※更新テーブルが複数ある場合、対象全てのテーブルに対し更新が掛かっている事、更新対象項目のうち主要項目８０%以上項目定義書の通りに更新が掛かっている事。	●	1	●
				帳票	異常終了する。正常終了するがPDFが作成されない。 ※出力帳ページのうち８０%以上出力されている事。 出力項目のうち８０%以上項目定義書の通りに出力されている事。			
				CSV TXT	異常終了する。正常終了するがCSV、テキストが作成されない。 ※出力帳レコードのうち８０%以上出力されている事。 出力項目のうち８０%以上項目定義書の通り出力されている事。			
				画面	アベンドする。動かない。データが全く登録(更新、削除）されない。 ※表示項目のうち８０%以上出力されている事。DBへの更新対象項目のうち、８０%以上項目定義書の通り主要項目に更新がかかっていること			
	パフォーマンス問題	処理完了までに長時間掛かり、処理結果を全く確認できない	②			●	1	●
		処理完了までに長時間掛かるが、処理結果を確認できる	③			任意	2	●
	設計ミス、ＰＧミス	同一LOT内の別機能に影響があり、その別機能の処理結果が全く確認できない。	④	バッチ	データ登録不正			
				画面	データ登録不正			

図 93-9 修正の優先判定基準例

（3） 現新比較検査

JBCC アジャイル開発では、現行プログラムでの実行結果と新プログラムでの実行結果を比較しながら機能を深堀し（潜在機能を顕在化し）、プログラムの完成度を高めていく。実行結果とは画面／帳票の出力内容やデータベースの出力内容を意味する。

特にバッチの機能の場合は、現新比較をすることで、当初想定していたデータパターンより多くの潜在データパターンをあぶり出せる。新システム構築において最も怖いのは、新機能の実現性よりも、現行機能のあぶり出し精度と同社では捉えている為、非常に有効な手段としている。但し、ToBe 型により開発する新システムにおいては、現新比較ができない為、別途の工夫が必要となる。

（4） パフォーマンス検査

プロトタイプ局面から、想定されるデータベースの最大件数を用意してテストすることで、早い段階でおおよその処理時間を体感しておき、後半になってパフォーマンスの問題に直面することがないようにしている。

（5） お客様確認

JBCC アジャイル開発では５回のイテレーションで機能を完成させることになるが、そのうちの２回目と４回目はお客様自身に確認を委ね、自身の目線で見極めしてもらうよう工夫している。

なお、アジャイル開発に向くお客様、向かないお客様がいるのも経験してきた。例えば、上

司に報告書を見せる日を１週間後に予定していた場合、１週間後に初めて上司に見せるタイプの人と、1 週間後に向けて、途中経過を都度都度上司に見せ、方向性を合わせていくタイプの人がいる。

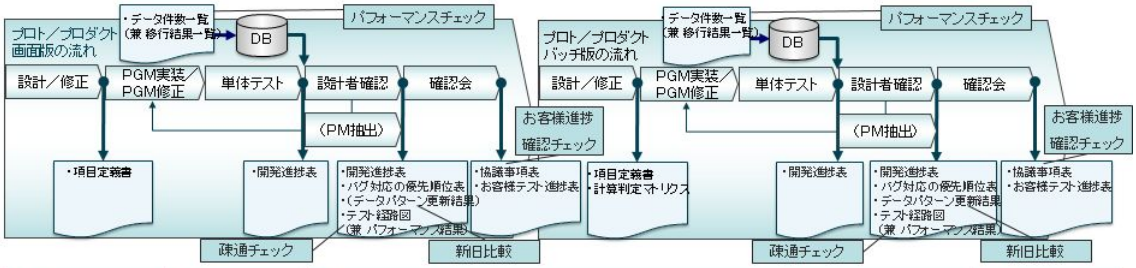
これは両者（お客様／開発ベンダー）共に言えることだが、都度の方法を取ることが出来なければアジャイル開発はなかなかうまくいかない。

4.4. 対策

２０１６年度のプロジェクト検査による分析結果を基に、２０１７年度はその対策として「裏どり検査」をスタートさせた（図 93-10）。

前述の通り、これまでのプロジェクト検査は紳士協定の下（プロジェクトメンバーが虚偽報告（誤報告を含む）を行わないという前提の下）、口頭確認で実施してきた。それに対し、裏どり検査は、プロジェクトを進めていく上で自然発生的に作られていくドキュメント（管理資料）を基に JBCC アジャイル開発の定石がきちんと守られているかを具体的に確認するものである。（このドキュメントの登録内容に虚偽（誤報告を含む）があるか否かは、現段階では検査の対象外としている）

この結果、２０１６年度の検査ではなかなか見つけられなかったプロジェクトからの虚偽報告（誤報告を含む）を浮き彫りにできることとなった。



テーマ	目的	前提	資料名
疎通チェック	疎通精度UP 網羅精度UP 修正見通し精度UP	・（ロット内）Aランクの課題に対して、修正基準に基づいて内部優先順位列に優先順位が入っていること ・単体テスト完了日、受入テスト完了日を管理していること ・（ロット間）ロットをつなげた状態のテストシナリオ ・テストシナリオに対する進捗状況	協議事項表 テストシナリオ
新旧対比	仕様漏れの回避	・（パッチ）計算判定マトリクス（データマトリクス）を基にした、データパターン数のセッション前後のアップデート表 ・（画面）スクリーンショット、データでの比較・・・等、画面の新旧対比をした結果がわかるエビデンス	マトリクス増減表
お客様進捗チェック	障害収束	・お客様進捗表が進捗していること	お客様テスト進捗表
パフォーマンス チェック	パフォーマンス問題の 早期発見	・機能要件時にお客様と合意した MAX データの想定件数 ・実際にプロト・プロダクト環境に移行したデータの件数がわかるもの ・パフォーマンスの測定結果	移行対象 データ一覧 機能別処理時間一覧
その他	負荷把握 実装確認	・協議事項表と開発進捗表のABCの打上数、対応残数にずれがないこと ・前回のセッションで上がったAランクの課題が、次回セッションまでにお客様確認欄に○がついていること ・積展開が必要な打上項目は、機能一覧に○がついていること ・開発後ランクに変更があったものは、機能一覧に補正後列を追加し管理されていること	協議事項表 開発進捗表 機能一覧表

図 93-10 裏どり検査の流れ

5. 達成度の評価、取組みの結果

5.1. 過去 3 年間の実績

JBCC アジャイル開発２０１４年１２月～２０１７年１２月の３年間での実績は、大規模（１億以上）１０件、中規模（３千万～１億）６件、小規模（３千万未満）２７件、現在進行中１７件の状況である（図 93-11）。

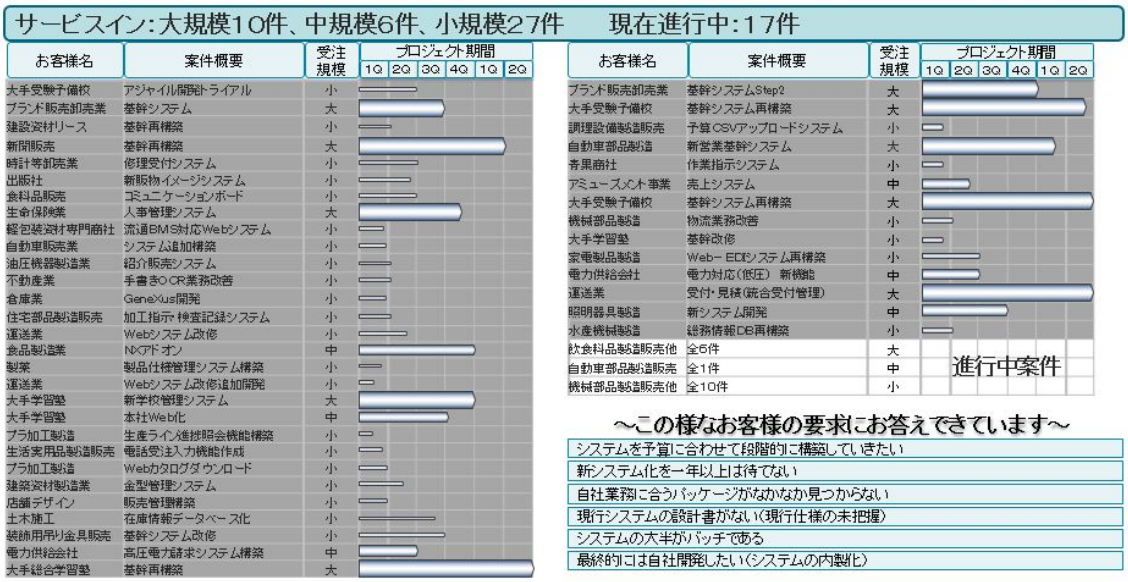
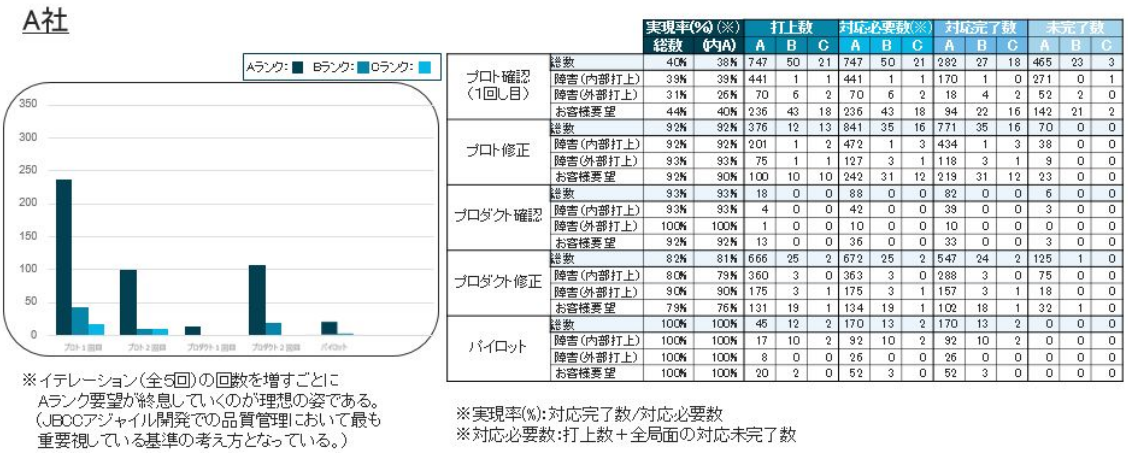


図 93-11 JBCC アジャイル開発のプロジェクト実績

同社の事例ではほぼ全ての契約について請負契約の形態で実施している。

5.2. 参考 1

お客様要望 A ランクの発生推移例（図 93-12）。



		実現率(%)(※) 総数	実現率(%)(※) (内A)	打上数 A	打上数 B	打上数 C	対応必要数(※) A	対応必要数(※) B	対応必要数(※) C	対応完了数 A	対応完了数 B	対応完了数 C	未完了数 A	未完了数 B	未完了数 C
プロト確認（1回し目）	総数	40%	38%	747	50	21	747	50	21	282	27	18	465	23	3
	障害（内部打上）	39%	39%	441	1	1	441	1	1	170	1	0	271	0	1
	障害（外部打上）	31%	26%	70	6	2	70	6	2	18	4	2	52	2	0
	お客様要望	44%	40%	236	43	18	236	43	18	94	22	16	142	21	2
プロト修正	総数	92%	92%	376	12	13	841	35	16	771	35	16	70	0	0
	障害（内部打上）	92%	92%	201	1	2	472	1	3	434	1	3	38	0	0
	障害（外部打上）	93%	93%	75	1	1	127	3	1	118	3	1	9	0	0
	お客様要望	92%	90%	100	10	10	242	31	12	219	31	12	23	0	0
プロダクト確認	総数	93%	93%	18	0	0	88	0	0	82	0	0	6	0	0
	障害（内部打上）	93%	93%	4	0	0	42	0	0	39	0	0	3	0	0
	障害（外部打上）	100%	100%	1	0	0	10	0	0	10	0	0	0	0	0
	お客様要望	92%	92%	13	0	0	36	0	0	33	0	0	3	0	0
プロダクト修正	総数	82%	81%	666	25	2	672	25	2	547	24	2	125	1	0
	障害（内部打上）	80%	79%	360	3	0	363	3	0	288	3	0	75	0	0
	障害（外部打上）	90%	90%	175	3	1	175	3	1	157	3	1	18	0	0
	お客様要望	79%	76%	131	19	1	134	19	1	102	18	1	32	1	0
パイロット	総数	100%	100%	45	12	2	170	13	2	170	13	2	0	0	0
	障害（内部打上）	100%	100%	17	10	2	92	10	2	92	10	2	0	0	0
	障害（外部打上）	100%	100%	8	0	0	26	0	0	26	0	0	0	0	0
	お客様要望	100%	100%	20	2	0	52	3	0	52	3	0	0	0	0

図 93-12 お客様要望 A ランクの発生推移（左）、局面別対応状況（右）

同社の事例では同型において、JBCC アジャイル開発での機能決定事項数＞ウォータフォール開発での機能決定事項数の関係で対応できており、今後の期待を増している。

5.3. 参考 2

パッケージ・アドオンとの対比事例（同機能程度での開発を、一方は某社パッケージ・アドオン、一方は JBCC アジャイルで開発）（図 93-13）。

図 93-13 パッケージ・アドオンとの対比事例

5.4. 参考 3

ウォータフォール開発と比較したとき（社内事例対比）において開発期間は平均 40%短縮できている（図 93-14）。なお、この短縮幅は、開発規模が大きくなるほど顕著なものになっている。

例えば図 93-11 中のある開発案件において、他社メーカーが 4 年の開発期間で見積もりしていたものに対し、同社では 1 年半で実現している。

図 93-14 ウォータフォール開発とのスピード比較

6. 今後の取組みと考察

6.1. お客様の直接の声

過去 3 年間の JBCC アジャイル開発の中でいただいたお客様の直接の声をご紹介する。

(1) 良かった点

- 見えるということがとても大きい（安心、期待)。
- 実際使用する画面を見ながらセッションを進めることで分かりやすい（紙には戻れない)。
- 現行仕様を把握しきれていない、現行システムの設計書がないといった場合は有効と感じる。
  （やるべき内容が見え隠れしてはいるので、アジャイルで深掘りしながら進められる）
- 段階分けした構築が大事。最初のシステム規模が大きなポイントとなる。
- 要望に対して柔軟な対応をしてくれる。
- 本番データを使用しながらのため、不備に気づきやすい。
- システム構築=プロジェクトに慣れていないお客様には特に向くのではないか（要件の線引き)。
- 難易度のある部分の検証で計算判定マトリクスを使ったＷｈｉｔｅＢｏｘテストを実施して頂いたので、仕様の漏れや考慮すべき課題も解決できて、安心できた。
- 同時並行していた他社開発の別システムがトラブル発生で進捗遅れがあった（同規模程度をパッケージアドオンで開発)。
- とにかく速く、週初めの要望が週末に見れる。手組では不可。
- こちらのやりたい事がほぼ実現できた。特に２回し目での変化（機能の変わりよう）が劇的で感動した。
- ＳＥと我々が密になってセッションを行えるかが一番大事と感じた。
- セッションの中でお互いの業務知識や開発手法を一致させられた。
- 他の大手ベンダーでは一定以上の要望は一切受け付けないが、要望に対して、柔軟に吸収してくれた。

(2) 悪かった点

- セッションへの参画が多い（拘束時間が多い)。
- 動作確認を自分たちでする必要がある。
- データ準備に時間がかかった。
- ウォーターフォール開発では、「バグが発生する度に、停止して対応する」という考え方だったが、JBCC アジャイル開発では、「機能全体を深堀りしてから品質を詰める」という考え方をとっている。(ちなみに、JBCC アジャイル開発ではバグを背負った形で進めている（P4「（１）スケジュールの固定化」参照のこと))。

6.2. 本手法の採用に至らなかった案件

JBCC アジャイル開発の採用に至らなかった案件の主な理由は以下の通りである。

(1) 画面デザイン

- 開発ツールで生成される画面の表現力が好みに合わない。

(2) 設計書

- 設計書を完全な物に仕上げてから開発を進めたい。

(3) コスト

- パッケージを採用したほうが安価で安全。
- JBCC アジャイル開発程の品質規格でなくてよいのでもっと安く仕上げたい。

(4) メーカー対応

- 開発ツールが国産ではない為、豊富な支援を受けられないのではという不安がある。

6.3. 今後の取組み

(1) 内部アジャイル

JBCC アジャイル開発では、開発局面におけるお客様間とのやり取りにおいては、ドキュメントレスを実現することが出来た。(但し、最終納品物として、設計書をまとめて提示するルールにしている)

しかし、それに伴う内部開発においては未だドキュメントを使用してプロジェクトを進めている。

そこで、今後は内部開発においてもドキュメントレスを目指していくこととしている。お客様との間では５回のイテレーションで機能を完成させるが、内部開発においては、設計者と実装者間で複数回のイテレーションを行い、機能を完成させることで今後取り組んでいく（今後はプログラム実装者が最終的なまとめとして設計書を書く事で考えている）(図 93-15)。

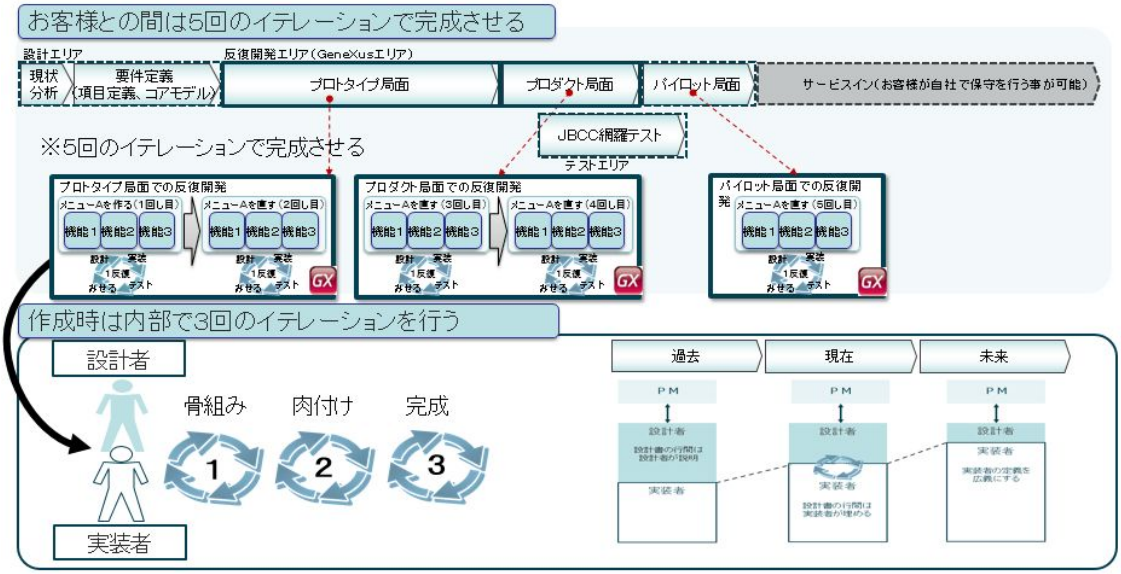


図 93-15 内部アジャイルイメージ

（2） アジャイル開発セッションでの音声認識（AI 組み込み）

JBCC アジャイル開発ではアジャイル開発セッションごとでの取り決めについて、協議事項表（議事録）に即日転記している。今後はこの協議事項表（議事録）そのものをシステム化（WEB 化）し、そこへの転記を音声認識（AI 組み込み）で対応していくことで現在取り組んでいる。