一般論文

統合設計情報システム開発でのアジャイル手法の適用

三田事業所　開発部　開発第3課
中西 一郎

1．まえがき

当所では三菱電機(株)三田製作所(以下、三菱電機)の統合設計情報システム(以下、TIPS：Taden Integrated Portal System)のソフトウェア開発及び保守を行っている。TIPSは、製品設計に関する様々な情報を管理する、設計部門・品質保証部門・工作部門・営業部門・生産技術部門向けのシステムである。一般的にはPLM(Products Lifecycle Management)と呼ばれる分野のシステムであるが、PLMベンダーが提供している他のシステムとは異なり、三菱電機の業務にフィットする様々なカスタマイズを可能にするシステムである。

TIPSは稼働から10年が経過し、働き方改革などの事業環境の変化、クラウドを代表とするITシステム技術の変化に伴い、これまで解決ができなかった課題への対応要求が高まった。一つ目はユーザー部門の要求であり、従来機能の改良などである。二つ目はセキュリティ担当部門からの要求であり、ORACLE社のJavaプラグインサポート終了に伴う、当プラグインに依存しないシステムへのUI再構築で、開発完了までの猶予期間は8ヶ月であった。この二つの要求に伴い、限られた期間に改良機能についてユーザー部門との合意形成を行いながら開発を進める必要があった。

この状況下、今回のソフトウェア開発に適した手法として、ユーザー部門との合意形成、改良を繰り返すことを前提とし、機敏かつ柔軟に開発するアジャイル開発を適用した。

本稿では、今回初めて適用し、目的に沿った成果が得られたアジャイル開発への取り組みについて紹介する。

2．システムの概要

TIPSはWebベースのアプリケーションソフトウェアであり、利用者はWebブラウザ上でシステムを利用する。

システム構成を図1に示す。

図1．TIPSのシステム構成

このTIPSは、デスクトップアプリケーションに近い表現力、操作性を実現する必要があるため、Webブラウザ上のJavaプラグインを利用していた。

まえがきで触れたユーザー部門の要求である従来機能の改良では、TIPS以外のシステムやモバイルデバイスとのUI統合を含めた抜本的な改良が必要となった。

Javaプラグインはセキュリティの問題があり、使用できる期限が決まっていたことから、Javaプラグインに依存しないHTML5を採用した、新しいUIプラットフォームへのシステム再構築を行った。

システム再構築の範囲は図2に示す通り、Webブラウザ上で動作する全ての処理と、アプリケーションサーバ上で動作する処理の一部となった。

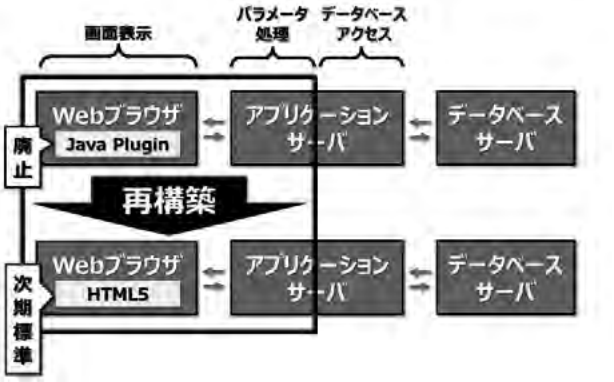

図2．システム再構築の範囲

3．ソフトウェア開発手法

今回のシステム再構築に適用するソフトウェア開発手法として、従来のTIPS開発で用いてきた「ウォーターフォール型開発」、そして2000年初頭に提唱された「アジャイル開発」の二つを適用候補とした。二つの開発手法にはそれぞれ特徴があるため、どちらがプロジェクトの品質・納期・コストをバランスよく達成するかは、開発プロジェクトの特性を十分に考慮し選択する必要があった。

ウォーターフォール型開発の長所として、「基本的に全体スケジュールが一旦決まれば変更はされないので、納期の見通しを立て易い」、「工程ごとの成果物作成とレビューによる品質担保」などがある。

一方、アジャイル開発では、ユーザー部門との合意形成と改良を繰り返すことを前提とし、要求の変化への対応を重視するプロジェクトに向いている。

4．アジャイル開発への取り組み

4.1　アジャイル開発の適用

3章で述べた通り、従来はウォーターフォール型開発を行っていた。図3にウォーターフォール型開発をV字モデルで示す。(a)要件定義は三菱電機が担当し、(b)外部設計から(g)システムテストまでの工程を当所が担当した。

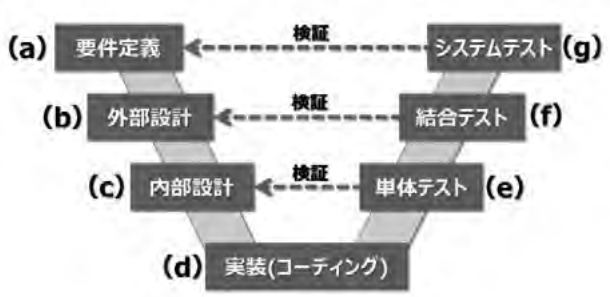

図３．従来のシステム開発（V字モデル）

今回のシステム再構築においては、UIの変更など要求仕様が最終形ではない前提で、プロトタイプを作成しユーザー部門と合意形成を行いながら進める必要があったため、ソフトウェア開発手法として従来のウォーターフォール型開発ではなくアジャイル開発を適用した。

4.2　アジャイル開発とは

アジャイル開発では、実現すべき機能、取り組むべき課題を個別に細分化したバックログという単位で、「計画→開発→レビュー→振り返り」を1セットの工程として、サイクルを回すことでバックログを消化し開発を進める。“アジャイル“とは「素早い」、「俊敏な」という意味であり、ユーザーへのリリースまでの時間を短縮できることを謳っている。アジャイル開発の代表的な用語を表1に示す。

今回は、アジャイル開発の中でも代表的な手法である「スクラム」を適用した。ラグビーのスクラムが語源であり、開発チームが一体となって働き、最短で製品の完成を目指す。スクラムの進め方を図4に示す。

表１．アジャイル開発の用語説明

用語	概要
バックログ	実現すべき機能、取り組むべき課題を個別に細分化したもの。
スクラム	・アジャイル開発の代表的な手法の一つ。 ・他の手法にエクストリームプログラミング(XP：Extreme Programming)やユーザー機能駆動開発(FDD：Feature Driven Development)がある。
開発チーム	バックログを元に動くソフトウェアを開発するメンバー。
スプリント	バックログが集まった機能単位で「計画→開発→レビュー→振り返り」を、短期間のサイクルで何回も繰り返すことで開発を進める。(短距離走の例え)
スプリント計画	バックログの優先順位付けや選定を、スクラムマスターを中心にチームで行う。
スクラムマスター	開発チームのリーダー。チームのパフォーマンス最大化に集中する。
デイリースクラム	毎日の進捗や問題を各自報告し、チームで共有する。
スプリントレビュー	実際に動くソフトウェアによるデモンストレーションを行いレビューする。
振り返り	完了したスプリントのプロセスを振り返り、以降のスプリントでの改善対策内容を検討する。スプリントレトロスペクティブと呼ばれる。

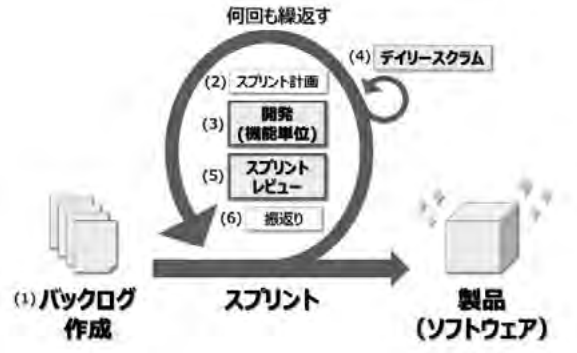

図４．アジャイル開発（スクラム）概略

4.3　アジャイル開発の実践

図4に示すアジャイル開発の実践を、流れに沿って以下(1)～(6)に示す。アジャイル開発においては表2に示すツールを活用した。いずれもWebベースのツールで、相互にデータ連携しているため、スプリントを効率よく回すことができた。

表２．使用したアジャイル開発支援ツール

名称	用途
JIRA	プロジェクト管理
Git	プログラムのバージョン管理を行う分散型ソースコード管理システム(オープンソースソフトウェア)
Bitbucket	上記GitをWebブラウザ上で操作するためのシステム。プログラムの変更点をチームで共有しコードレビューも可能。
Confluence	ドキュメント管理

(1)バックログの作成

アジャイル開発の最初には、作業内容の概要としてバックログを作成する必要がある。TIPSを構成する全ての画面を分析し、バックログを作成、JIRAに登録した。最終的にバックログ数は合計958件になった。バックログは開発途中に発生する問題等を含めて随時追加した。

実装作業の観点から、バックログの粒度レベルをどう定めるか、チーム全体で議論を行った。その結果、画面表示やボタン操作の処理を一つのバックログの単位として登録することに決定した。粒度を均等化したことで作業単位が整理され、工程の計画精度が向上した。

作成したバックログの情報は、スプリント計画を立てる際に使用した。

(2)スプリント計画の立案

スクラムチームは図5に示す通り、スクラムマスター及び2チーム各5名体制で編成した。

スプリント全体計画は第1回のスプリント開始前にスクラムマスターが立案した。リリース予定日、及び1スプリントを実働10日として、10回のスプリントを実施する計画とした。各スプリントでは、終了時に振り返りと次のスプリント計画を実施することとした。

スプリント計画に従い実施するスプリントで、各チームは受け持ったバックログに優先順位付けを行い、それぞれのスプリントでどのバックログを消化するか、スクラムマスターを中心にチームで検討を行った。

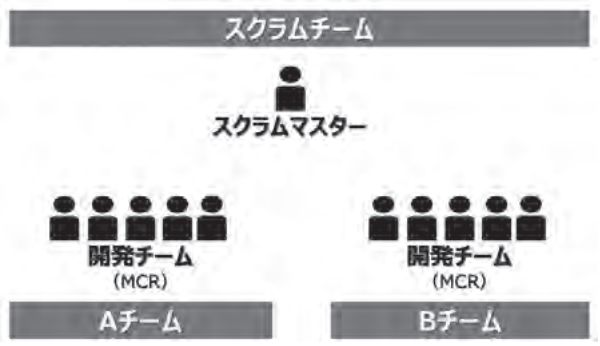

図５．スクラムチーム

(3)開発(機能単位)と使用ツール

(a)実装

バックログの内部設計・コーディング・デバッグを実施した。該当するスプリントに割り当てたバックログ実装作業のステータス管理はJIRAのカンバンボードで行った。

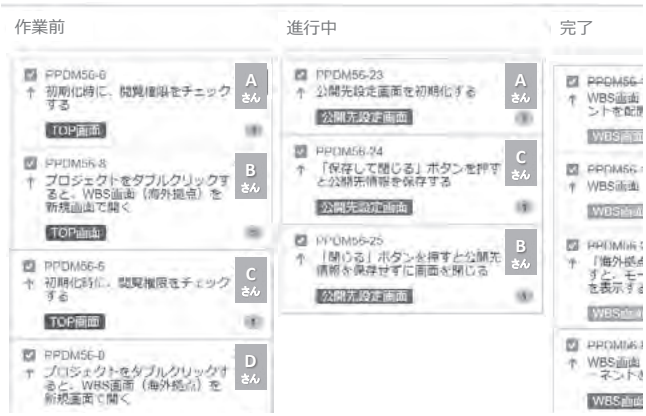

図６．JIRAのカンバンボード

図6に示すJIRAのカンバンボード画面にて、個々のバックログはアイコンで視覚化される。

開発メンバーは、各バックログのアイコンを「作業前」、「作業中」、「完了」などの領域にドラッグ&ドロップ操作することで、直感的に作業ステータスを更新することができた。JIRAのカンバンボード活用により、スクラムマスターは、開発メンバーのバックログ作業状態を容易に把握し、タイムリーなフォローが可能になった。

(b)ソースコードの管理とレビュー

バックログ単位でのコーディング・デバッグが完了すると、作成したソースコードをGitに格納した。

格納したプログラムを本番プログラムにマージ(統合処理)する前に、BitbucketをWebブラウザ上でのコードレビューを実施した。これにより、プログラム変更部分が容易に識別できるようになったため、

効率化と品質を確保することができた。

(c)ドキュメントの作成と管理

従来は開発が進むにつれ、作成する文書が増え、文書を探す作業に時間を要する問題があった。そこで仕様書や議事録などの作成は、WordやExcelではなくConfluenceを活用した。Confluenceで作成した文書は全文検索機能を用いてキーワード検索することで、確認作業の効率化が図れた。

(4)デイリースクラム

スプリント期間中は毎朝デイリースクラムを実施した。前日の実施結果、当日の実施予定、現在課題となっていることを各メンバーが手短に口頭で報告した。毎回15分程度で終わらせるが、実装作業の行き詰まりがデイリースクラムにより見通しが立つこともあった。

しかし報告や相談の内容が詳細になり、デイリースクラムが長引く傾向にあった。その場での問題解決ができそうにない場合は、別途打合せの場を設けることのルールを守ったことで、デイリースクラムにメリハリをつけた。

進捗管理では、図7に示すバーンダウンチャートを用い、バックログ消化状況の計画に対する乖離の有無と度合いをビジュアル的に把握した。これを毎朝チェックする運用とすることで、チームのバックログ消化に対する意欲が高まった。

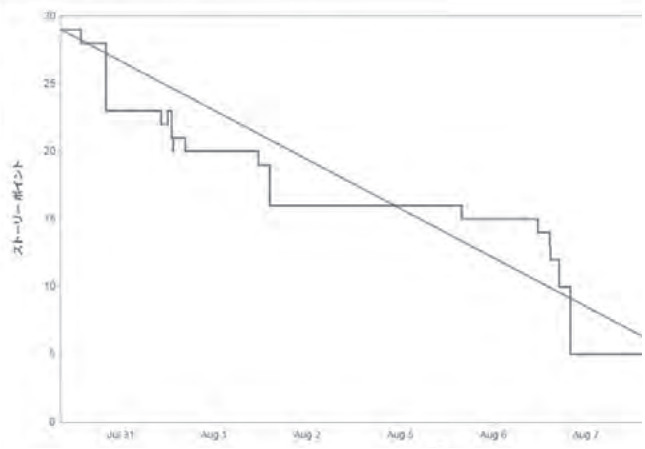

図7．バーンダウンチャート

(5)スプリントレビュー

スプリントの最後にスプリントレビューを実施した。スプリントレビューでは実際に動くソフトウェアを用いてチーム全員がユーザー目線でチェックを行った。デモンストレーション中に不具合が発見されたり、改善事項が提案されることがあれば、新たなバックログとしてJIRAに登録し、以降のスプリントで対応した。

アジャイル開発では、各スプリントレビュー時に問題の検出が可能となるため、機能ごとに早く是正することでフロントローディング化できた。

スプリントレビューは限られた時間の中で実施するため、全ての実装仕様と、その中で今回デモンストレーションを行う仕様をドキュメントで提示した後にデモンストレーションを行うよう工夫した。これにより仕様漏れが無いかチェックが可能となり、品質を確保できた。

(6)振り返り

今回はKPT(Keep/Problem/Try)手法を用いて振り返りを実施した。KPTではKeep(継続したいこと)、Problem(問題)、についてチームメンバーで振り返りを行い、Try(改善策)を導き出した。このKPT手法の適用により、スプリントを繰り返すごとに作業効率を上げることができた。KPTの内容は振り返りの場で思い出すのではなく、開発作業の中で都度Confluenceにメモしておくことで、効率よく振り返り活動が進行した。

4.4　アジャイル開発の実践での課題と対策

アジャイル開発の実践を通して直面した課題とその対策内容を以下に示す。

(1)開発技術の共有

アジャイル開発では、各スプリントの中で実際に動くソフトウェアの作成が必要で、プログラムに直接関わる問題の解決が求められる。今回は新しいUIプラットフォームに依存したプログラム上の問題が多く、インターネットや書籍による情報も少ない。このため、プログラム作成のノウハウ共有が必要となった。対策として、各自が保有するUIプラットフォームのプログラム作成ノウハウをQ&A形式でConfluenceに掲載し、開発メンバーと共有した。その結果、開発メンバーはそのQ&Aを確認することで効率的にプログラム作成の問題を解決し、実際に動くソフトウェアを限られた期間で作成することが可能となった。

(2)機能・画面の連携対応

バックログで設計・製作したプログラムは、スプリントレビューで動くプログラムとする必要があり、各機能の結合または仮結合が必要となる。各画面間の遷移処理についても同様である。機能や画面間での受け渡しで必要なパラメータの不足が判明し、結合する際になってプログラム修正が必要となる事態が、開発当初に発生した。この問題に対応するため、機能・画面全体の統合イメージの早期確立が課題となった。対策として図8に示すUI検討チームを新設し、各機能の実装作業と並行して、機能や画面の統合イメージを検討した。この結果、バックロ

グで設計･製作する作業の中で各機能や画面のつながりの把握が可能となり、呼び出し元画面が呼び出し先画面に送信が必要なパラメータなどのインタフェース仕様を、バックログの作業段階でまとめることができた。

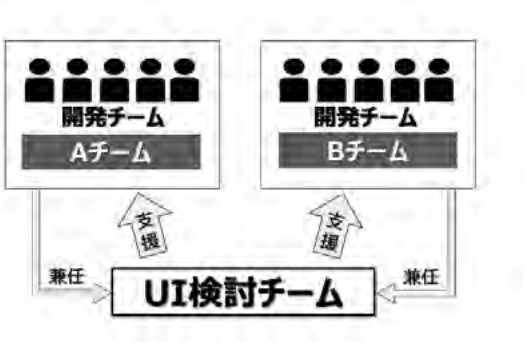

図８．UI検討チーム

(3)類似するクラス･メソッドの抑制

複数メンバーで同機能のバックログの実装に着手した場合、同じような処理を行うプログラムのクラスやメソッドが異なる名称で複数作成される問題があった。対策として、クラス図を全体で共有し、デイリースクラムでは、作成するクラスやメソッドの詳細情報を共有する運用とした。

その結果、プログラムのクラスやメソッドの乱立を抑制し、プログラム流用による効率化を図ることができた。

(4)プログラム競合回避

今回は、機能単位での開発を繰り返すアジャイル開発の特性もあり、特定の機能を複数メンバーで分担して実装する場合が多かった。作成したプログラムをGit(バージョン管理システム)に登録する際に、開発メンバー間でプログラム編集の競合が発生した。競合する両者のプログラム変更箇所によっては、マージは手動で行う必要があり非効率であった。このため、開発メンバー同士のコミュニケーションが課題となった。対策として、デイリースクラムにてプログラムの編集状況を共有した。その結果、競合を事前に回避し、非効率な手動マージを最小化することができた。

4.5　アジャイル開発導入成果

実際に動くソフトウェアを作成し、ユーザー部門との合意形成と改良を繰り返す中で、柔軟で効率的な開発が可能となった。今回は抜本的なシステム再構築であったが、各スプリントを計画通り実施し、短期間でシステム再構築を完了できた。

ツール活用や、デイリースクラムをはじめとした日常的コミュニケーションによる生産性向上も実感することができた。

５．今後の課題

今後も機敏かつ柔軟な開発を推進するために、アジャイル開発の運用のルール化を行い、アジャイル開発を定着させる必要がある。アジャイル開発で課題となる繰り返しテストを自動化し、さらなる品質向上を検討する。

６．むすび

本稿で紹介したアジャイル開発を導入したことで、従来は全体の設計→全体の実装→全体の試験の流れであった開発が、実際に動くソフトウェアを機能ごと作成し、ユーザー目線での改善や作り込みをこまめに行うスタイルに変わった。その結果、システム再構築の早期リリースを実現することができた。

また、課題の視覚化とチームワークがシステム開発の生産性向上に役立つことを、本開発作業を通して実感することができた。

今後、アジャイル開発の成功事例を積み上げ、適用範囲の拡大を図る。

最後に、本活動に当たり多くの助力をいただいた関係者各位に深く感謝申し上げる。

執筆者紹介

中西　一郎　　ナカニシ　イチロウ
1998年入社。主に生産情報システムのソフトウェア開発に従事。現在、三田事業所開発部開発第3課。