XDDP と SCRUM によるチーム型派生開発の取り組み事例

The proposal of Team-Based software derivative development by XDDP and SCRUM

ソニーイーエムシーエス株式会社 湖西サイト 設計１部
Sony EMCS Corporation Kosai Site Design Engineering Dept.1
勝又 淳
Atsushi Katsumata

Abstract In order to survive ever-intensifying product competition these days, one of our most pressing needs is to meet the growing demand for quicker delivery of higher-performance / -quality and lower-cost embedded devices. However, our human, physical and financial resources are not necessarily abundant due to the prolonged economic downturn - in particular, the resources that can be allocated to derivative development is more limited compared with those allocated to new product development. To respond to such a harsh environment surrounding the derivative development, we will adopt approaches from two different aspects - technical aspect (XDDP) and human management aspect (SCRUM), which will allow us to develop higher-quality derivative embedded software at lower cost with less human resources.

1. はじめに

1.1 概要

昨今の組み込み機器は，製品競争の激化により，高機能，高品質(Quality)，低コスト(Cost)，短納期(Delivery)への要求が強く，これらの課題解決が急務となっている．しかしながら，長きに渡る景気低迷によりヒト・モノ・カネといったリソースが潤沢に割かれるわけではない．派生開発においては，新規開発に比べてその傾向がさらに強い．そこで，派生開発において，迅速かつ最適な人員で高品質なソフトウェアのチーム開発を実現するために，派生開発に特化した開発プロセス的なアプローチ(XDDP)とヒューマンマネージメント的なアプローチ(SCRUM)の二つの側面で課題解決に取り組む．

1.2 課題

弊社は，プロフェッショナル用放送機器の製造拠点という位置づけではあるが，設計部門が存在し，商品設計業務を行っている．

我々の組織では，事業本部で設計された組み込み機器の派生モデルの開発案件が多い．そのため，派生開発を短納期・高品質・低コストで行うことがビジネス課題の一つであり，改善施策を日々模索している．

開発プロジェクトでは，機械，電気，ソフトウェア，テストの各グループに分かれて派生開発を行っている．ソフトウェアグループに属している我々は，厳しいリソース制約の中でも，迅速かつ最適な人員で高品質なソフトウェアの開発を実現することで，プロジェクト全体のQCD改善(高品質<Quality>，低コスト<Cost>，短納期<Delivery>)に貢献できると考えている．それを実現するために，現状の課題を三つあげる．

ソニーイーエムシーエス株式会社 湖西サイト 設計１部
Sony EMCS Corporation Kosai Site Design Engineering Dept.1
静岡県湖西市境宿 554 TEL：053-577-4329
554, Sakaijyuku, Kosai-shi, Shizuoka, Japan

(1) 後戻りに関する課題

組み込み機器の開発では，ハードウェアの開発プロセスとソフトウェアの開発プロセスの実施時期にずれがあることが多い．ハードウェアに起因する不具合が，プロジェクトの後半で発生すると大きな後戻り要因となるので，ハードウェアに起因する不具合は，できるだけ早期に排除する必要がある．そのためには，ハードウェア検証を行うためのソフトウェアが必要である．

本来は，ハードウェア開発チーム内で検証用のソフトウェアを用意すべきだが，リソースの関係上，その実現が難しい場合が多い．そのため，ソフトウェア開発チームが，製品適用ソフトウェアとは別にプロトタイプという形でハードウェア検証用のソフトウェアを提供することが多い．当然，そのプロトタイプの開発には，製品適用ソフトウェアとは別の開発コストが必要となる．プロトタイプの開発が重複作業となっているので，製品開発と一本化して開発コストを抑えたい．

また，ハードウェアの変更による問題が，プロジェクトの後半で検出されると多大な後戻り工数が発生するので，そのリスクを回避するためにも，ハードウェア検証の際には，プロトタイプではなく，製品に近い品質の高い状態でのソフトウェア提供が望ましい．

(2) ソフトウェアを開発するチームに起因する課題

我々の組織が着手する組み込み機器では，５～１０名程度の人員で組み込みソフトウェアの開発を行うことが多い．毎回同じメンバでチームを構成するとは限らないので，チームビルディングが非常に重要である．

開発プロジェクトの早期からチームビルディングに着手し，チーム内の意識・モチベーションをプロジェクトの始めから終わりまで高いレベルで保つことが必要である．また，コミュニケーション不足などによる人的問題を排除し，知識共有を効果的に行うことも重要な課題の一つである．複数人のチームでソフトウェアの開発を行うので，ゴールに向かって効率良く確実に進んでいけるチーム作りが必要である．

(3) ソフトウェアのリリースタイミングに起因する課題

過去のプロジェクトにおいて，プロジェクトの後半に突発的なデモンストレーションの要求や製造ラインでのテスト稼働要求が度々あった．デモンストレーション要求や製造ラインのテスト稼働要求は，ビジネスに直結することも多いので，できるだけ迅速に応えたい．プロジェクトの後半では，このような突発的なリリース要求に対しても，製品レベルの品質を保持したソフトウェアを迅速に提供できるような工夫が必要である．

2. 適用する手法と期待効果

2.1 XDDP

XDDP(eXtreme Derivative Development Process)とは[1]，ソフトウェアコンサルタントの清水吉男氏によって考案された「派生開発に特化して合理化された開発プロセス」である．XDDP の主な特徴は，追加と変更を別のプロセスに分けるところである．変更部分では，変更要求仕様書・トレーサビリティマトリクス（以下ＴＭと称する）・変更設計書をベースドキュメントとし，母体との差分に着目して徹底的に無駄を排除したプロセスである．XDDP では，組み込みソフトウェア開発，アプリケーション開発，システム開発などのジャンルを問わず派生開発であれば適用可能である．

2.2 SCRUM

SCRUM とは[2]，アジャイル開発手法の一つであり，その名の通りチームが一丸となって開発を進められるように，プロジェクトマネジメントの側面に焦点を当てた手法である．プロジェクトマネジメントを通じて，チーム全体の生産性向上を目指すという特徴がある．

SCRUM では，プロダクトバックログと呼ばれる開発案件リストから優先順位の高い順に 1 回の反復（スプリントと呼ぶ）で開発可能なものを選び，プロダクトバックログを減らしながら開発を進めていくものである．通常，一つのスプリントを 2 週間から 6 週間のタイムボックス(固定区間)に区切る．本論文では，1 スプリントを 2 週間のタイムボックスとしている．

SCRUM チームの役割は三つある．責任者として要件と優先度を決める「プロダクトオーナー」，プロジェクトがスクラムに沿って進むように下支えする「スクラムマスター」，開発者の集団であ

る「チーム」である．スクラムチームは，一般的に 5～9 人位で構成するといわれている．SCRUM に欠かせない重要なポイントは，チームメンバが自律的・主体的に動かなければならないことである．最初はスプリントの目標に基づき，その後は毎日のデイリースクラムを通じて，日次ベースで自ら組織的な調整を行わなければならない．SCRUM の概要図を図１に示す．

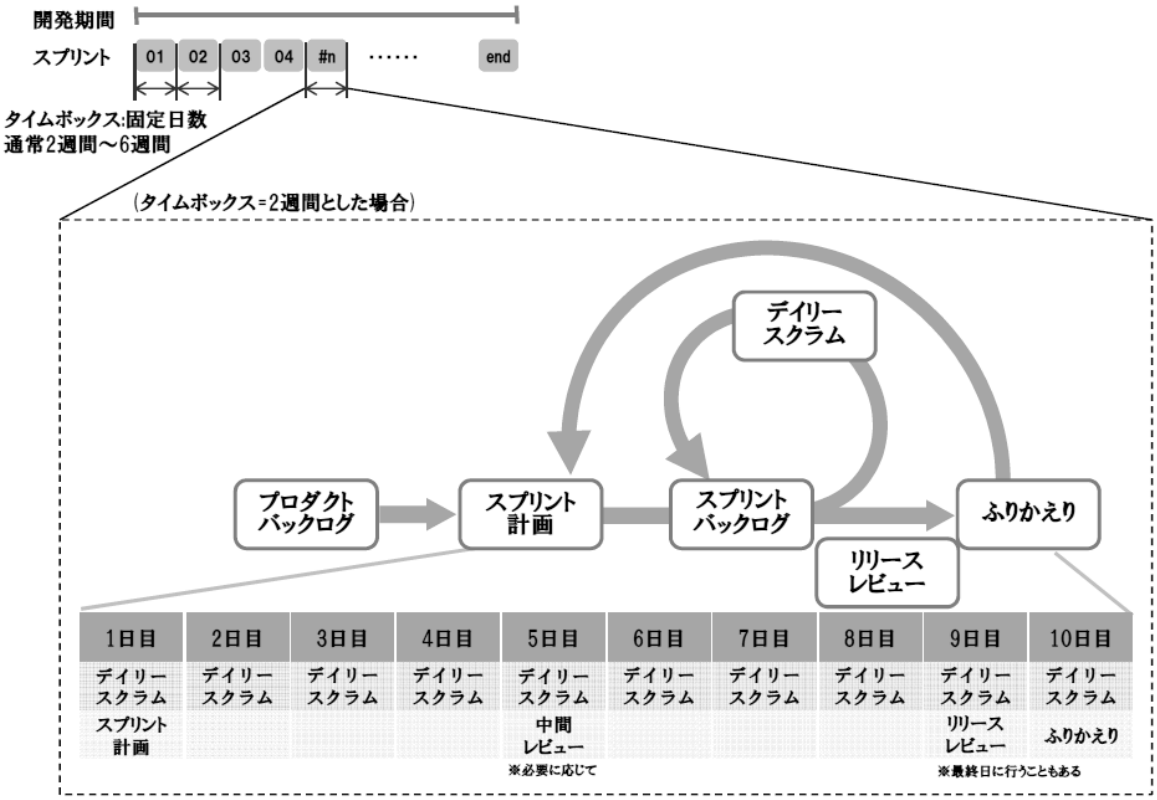

図１： SCRUM の概要図

(1)　スプリント計画

スプリント計画は，スプリントの初日に当該スプリントで行うスプリントバックログ（作業項目）の合意を図る． スプリント計画は２部で構成される． 前半で開発するプロダクトバックログを選び，後半で作業項目（タスク）を「スプリントバックログ」にまとめ，全タスクを消化できるようにチーム主体で計画を立てる．

(2)　デイリースクラム

SCRUM のプラクティスの一つに「デイリースクラム」がある．デイリースクラムは，毎日定刻に同じ場所に集まるスタンドアップ形式のミーティングである．デイリースクラムの最大の特徴は，最大でも１５分程度で終了するところである．全員が，昨日から今日までに行ったこと，今日これから行うこと，現状の問題点の３点についてのみ話をする．

(3)　ふりかえり

SCRUM のプラクティスの一つに「ふりかえり」がある．各スプリントの終了時に，KPT 形式でふりかえりを行う．KPT とは，Keep/Problem/Try を意味する．Keep は今回のスプリントで良かったことや次回のスプリントでも続けたいことをあげ，Problem は次回のスプリントでは改善したいことあげ，Try は Problem の具体的な解決策または新たに挑戦したいことをあげる．

2.3　手法の適用方法

(1)　XDDP の適用方法

一つ目の施策として，派生開発に特化して合理化された開発プロセスである XDDP を適用する．XDDP は，すべての変更要求のドキュメントを作成し，レビューを行い，十分に検討した後に一斉にコーディングを行うことで後戻りなく高品質なソフトウェアの開発を行うプロセスである． 本論文では，プロジェクトを前半・後半の二つのフェーズに分割し，ハードウェアの変更に伴う変更要求を対象とする前半に XDDP を適用する．XDDP の適用を前半に絞ることで，製品と同等の品

質状態のソフトウェアをハードウェア検証に用いることが可能となり，その結果，ハードウェア検証用のプロトタイプを必要とすることなく，ハードウェア要因の不具合を早期に排除することが可能となる．これにより，一つ目にあげたハードウェア検証に適用するソフトウェアの品質と後戻りに関する課題を解決に導く．

(2) SCRUM の適用方法

二つ目の施策として，チームの生産性向上に主眼を置いたマネジメント重視型のアジャイル開発手法である SCRUM を適用する．チームを構成するメンバの自律性を重視し，チーム一丸となって目標達成に向かって進むことで，チーム全体の生産性が向上することを期待する．これにより，二つ目にあげたソフトウェアを開発するチームに起因する課題を解決に導く．

また，SCRUM は，一定の期間で区切りながら動作するソフトウェアの開発を進める反復的な開発アプローチであるので，三つ目にあげたソフトウェアのリリースタイミングに起因する課題の解決にも適する．SCRUM を適用するのは今回が初めてであり，SCRUM に慣れるためにも，できるだけ多くのスプリントを繰り返したいという理由でタイムボックスを２週間に設定した．

3. 適用事例

3.1 適用プロジェクトの概要

本論文で提案する XDDP と SCRUM の適用プロジェクトの概要を表１に示す．業務用放送機器の派生モデルの開発案件であり，開発期間は約１年，ソフトウェア開発者７名程度の規模である．

表 1： 適用プロジェクトの概要

開発概要	業務用放送機器の派生モデル開発
期間	2011/04 ～ 2012/03(約１年)
人数	約 15 名　ソフトウェア(7 名)，ハードウェア，テスト，PM
母体行数	約 500(KLOC)
変更行数	約 50(KLOC)
スプリント数	24 回 (Timebox = 2 週間)

3.2 適用方法と工夫した点

本論文で提案する XDDP と SCRUM の適用プロジェクトの開発計画を図２に示す．

図２： XDDP/SCRUM 適用プロジェクト開発計画

１章であげた課題を解決するために，我々は開発期間の区切りかたとプロセスの適用方法を工夫した．組み込み製品の派生開発であるため，ハードウェアも変更対象となる．ハードウェアの開発プロセスに合うように，開発期間を大きく前半と後半の二つに分けた．

プロジェクトの前半(Sprint #1～Sprint #13)は，ハードウェア変更に伴うソフトウェアの変更（アーキテクチャの変更含む）を対象とし，プロジェクトの後半(Sprint #14～Sprint #21)は，新規機能の追加とそれに伴うソフトウェア変更を対象とする．(Sprint #22～Sprint #24)では，不具合修正及び設定値の微調整として割り当てる．プロジェクトの終盤には，各方面からのデモンス

トレーション要求や製造ラインからのテスト稼働要求などの突発的なリリース要求が多いため，Sprint #17 以降は，１スプリントを１イテレーション（反復型開発プロセスにおける一回の繰り返し）としている．

プロジェクトの前半(Sprint #1～Sprint #13)の主な作業内容と成果物を表２に示す．プロジェクトの前半では，アーキテクチャの変更を含むハードウェアに関する変更要求のみを対象とし，母体のソースコード解析と並行して変更要求仕様書を作成した後，トレーサビリティマトリクスと合わせてレビューを実施し合意を図る．変更設計書には，具体的なソースコードの変更内容と確認方法（単体テスト・結合テスト）を記載する．変更設計書のレビューを実施し，変更漏れや変更間違いなどの欠陥をコーディング前に除去する．レビュー完了後，一斉にソースコードの修正に取り掛かり，ハードウェア入手後に変更設計書に記載したテストを実施する．その後，テストチームによるシステムテストを実施する．このように，XDDP に沿った作業を行うことで，ハードウェア検証用のプロトタイプではなく，製品レベルの品質を確保したソフトウェアをハードウェア検証として適用することで，早期にハードウェアに起因する不具合を検出でき，プロジェクト終盤での後戻りリスクを抑制することができる.また,以前まで作成していたハードウェア検証用のプロトタイプが不要となるため，プロトタイプの作成工数削減にもつながる．

表 2：プロジェクト前半(Sprint #1～#13)における主な XDDP の作業内容と成果物

項番	成果物	作業内容
1	スペックアウトドキュメント	母体のソースコードの解析(リバースエンジニアリング)を行う.
2	変更要求仕様書	変更要求の内容及び母体のソースコードの解析結果より変更要求仕様書を作成する.
3	レビュー記録表１	変更要求仕様書のレビューを実施して変更仕様の合意を図る.
4	トレーサビリティマトリクス	要求仕様を行要素,ファイル名を列要素としたトレーサビリティマトリクスを作成し，要求仕様に衝突がないことを確認する.
5	変更設計書	変更要求仕様書を元に変更設計書を作成する.
6	レビュー記録表２	変更設計書のレビューを実施して早期に欠陥を除去する.
7	ソースコード	変更設計書に従って一斉にコーディングを行う.
8	テスト結果	変更設計書に記載している単体・結合テストを行う.

プロジェクトの後半(Sprint #14 以降)では，残りの変更要求と新たに追加となる機能を計画的に実装していく．後半では，基本的には１スプリント１イテレーションとし，機能ごとに設計・テストを進めていく．プロジェクトの後半で行う主な作業内容と成果物を表３に示す.

表 3：プロジェクト後半(Sprint #14～#21)における主な XDDP の作業内容と成果物

項番	成果物	作業内容
1	追加機能要求仕様書	追加機能要求（プロダクトバックログ）を元に追加機能要求仕様書を作成する.
2	変更要求仕様書	追加要求に伴う変更箇所の変更要求仕様書を作成する.
3	レビュー記録表１	追加機能要求仕様書,変更要求仕様書のレビューを実施し，要求仕様の合意を図る.
4	トレーサビリティマトリクス	要求仕様を行要素,ファイル名を列要素としたトレーサビリティマトリクスを作成し，要求仕様に衝突がないことを確認する.
5	関数設計書	追加機能要求仕様書を元に関数設計書を作成する.
6	レビュー記録表２	関数設計書のレビューを実施して早期に欠陥を除去する.
7	ソースコード	関数設計書に従ってコーディングを行う.
8	テスト結果	関数設計書に記載している単体・結合テストを行う.

一方，第２の課題としてあげているソフトウェア開発チームに起因する課題に関しては，プロジェクトの初期段階から SCRUM を適用することで解決する．SCRUM で行うプラクティスを表４に示す．SCRUM では，スプリントごとにリリースできるレベルのソフトウェアを提供するのが一般

的ではあるが，まだハードウェアが手元にないプロジェクトの前半から，生成するドキュメントを動くソフトウェアの代用として SCRUM のプラクティスを実施する．このように，プロジェクトの最初から最後まで一貫して SCRUM のプラクティスを実施することで，SCRUM のプラクティスに慣れることと，チームの意識・モチベーションを最後まで高いレベルで保つことを期待する．

表 4： プロジェクトを通して行う SCRUM のプラクティス

項番	プラクティス	概要
1	スプリント計画	当該スプリントの最初に行い，対象とするバックログに関する合意を図る．
2	デイリースクラム	毎日定刻に同じ場所に集まって 1 日単位のふりかえりを行う．
3	リリースレビュー	当該スプリントでリリースする成果物に関する説明を行う．
4	ふりかえり	当該スプリントの最終日に KPT によるふりかえりを行う．

4. 適用結果

XDDP/SCRUM 適用プロジェクトの適用結果及び未適用時との比較を表５に示す．

品質面に関しては，すべての機能実装およびテストが完了した正式版以降のソフトウェアに対して品質保証部が行う QA 検査での指摘不具合件数が，XDDP/SCRUM 未適用プロジェクト時には 9 件だったが，今回のプロジェクトでは 0 件と大幅に改善することができた．

生産性面に関しては，１時間当たりに記述するソースコード行数の平均値を表す生産性(LOC/人・h)が，XDDP/SCRUM 未適用プロジェクト時には 37.4(LOC/人・h)だったが，今回のプロジェクトでは，150.3(LOC/人・h)と大幅に改善することができた．

納期に関しては，従来も納期に遅れることはなかったが，XDDP と SCRUM の適用という新たな試みを行った今回のプロジェクトにおいても遅延することなく計画通り出荷することができた．

また，一つ目の課題で，プロジェクト後半でのハードウェア要因による後戻りリスクの回避をあげているが，その結果を表すハードウェア要因不具合検出件数を表６に示す．今回のプロジェクトでは，プロジェクト後半でのハードウェア要因による不具合の件数を 0 件にすることができた．

表 5： XDDP/SCRUM 適用結果（未適用時との比較）

プロジェクト	XDDP/SCRUM 未適用プロジェクト	XDDP/SCRUM 適用プロジェクト(今回)
プロジェクトの概要	［案件］業務用組み込み機器の派生モデル開発Ａ ［期間］2008/09 ～ 2009/05 （約 10 ヶ月） ［人数］約 12 名 （ソフトウェア 5 名） ［要求］約 50 件 ［母体］約 650KLOC	［案件］業務用組み込み機器の派生モデル開発Ｂ ［期間］2011/04 ～ 2012/03 （約 1 年） ［人数］約 15 名 （ソフトウェア 7 名） ［要求］約 70 件 ［母体］約 500KLOC
生産性 (LOC/人・h)	37.4 LOC/人・h	150.3 LOC/人・h
品質(正式版以降の QA 検査指摘不具合件数)	9 件	0 件
納期(遅延日数)	0 日（計画通り出荷）	0 日（計画通り出荷）

表 6： ハードウェア要因不具合検出件数

成果物	プロジェクト前半検出件数	プロジェクト後半検出件数
ハードウェア(基板)要因	7 件	0 件
ハードウェア(FPGA)要因	6 件	0 件
メカ要因	2 件	0 件

二つ目にあげたチームに起因する課題に関しては，プロジェクトの前半から一貫して SCRUM のプラクティスを適用することで，起きた問題に対してチーム内で自発的に改善策を検討し問題解決をする自己組織型チームへと成長する効果があった．

デイリースクラムの効果としては，毎日決まった時間にミーティングを行うことで，リズムが体に刻まれ，自然と定刻に同じ場所に集合するようになり，個々のリズムがチームのリズムと調和してくるようになった．問題が発生しそうになった際には，すぐに解決の場を設けることで，早期に問題発生を予防することができた．自分達の立ち位置を共有し，各自が安心して 1 日のタスクに取り組むことができたので，チームの生産性向上につながった．

ふりかえりの効果としては，スプリントごとにふりかえりを行うことで，改善意識とモチベーションを高く保つことができたことがあげられる．スプリントで区切ることにより，次も新たな気持ちで頑張ろうという気持ちになるので，チーム全体のモチベーション向上につながった．

三つ目にあげたモジュールのリリースタイミングに起因する課題に関しては，プロジェクトの後半では，1 スプリントを 1 イテレーションとしたことで，突発的なデモンストレーション要求や製造ラインからのテストラン要求に迅速に応えることができた．

5. 考察

現状における三つの課題を提起し，解決策として XDDP と SCRUM を適用した．その結果，現状の課題を解決し，QCD 改善の効果とチームの成長を得ることができた．以下にその論拠を述べる．

5.1 QCD 改善効果についての考察

４章の適用結果で述べたように，品質面，生産性面ともに改善効果があった．これはメンバ全員が XDDP を正しく理解して実践したことと，SCRUM のプラクティスの一つであるふりかえりを通じて，QCD を意識した継続的な改善に取り組むことができたためと推測する．

また，プロジェクトの前半で，ハードウェアに影響のある変更要求とアーキテクチャに関わる部分の変更要求に絞って XDDP のプロセスを適用したことで，製品レベルに近い品質の良いソフトウェアをハードウェア検証に適用することができた．その結果，プロジェクトの前半でハードウェア要因の不具合を除去することができたので，プロジェクトの後半では，追加機能の要求に集中して取り組むことができた．その結果，全体の QCD 改善の向上につながったと推測する．

5.2 XDDP と SCRUM の適用効果についての考察

XDDP と SCRUM の適用効果と不足要素を表７に示す．また，本論文で現状の課題としてあげた項目とその課題解決に有効だった手法を表８に示す．

このように，XDDP のみで適用する際の不足要素と SCRUM のみで適用する際の不足要素がそれぞれ存在する．XDDP と SCRUM を併用することで，それぞれの不足要素を補うことができ，改善効果につながったと推測する．また，今回のプロジェクトでは，XDDP と SCRUM の併用による負の作用がなかったことも要因の一つであると推測する．

表 7： XDDP と SCRUM の適用効果と不足要素

適用する手法	主な適用効果	不足要素
XDDP のみ適用	・QCD 意識(特に品質面) ・後戻り工数の削減	・チームの一体感 ・継続的な改善意識 ・定期的なリリース
SCRUM のみ適用	・チームの一体感 ・継続的な改善意識 ・定期的なリリース	・QCD 意識(特に品質面) ・後戻り工数の削減
XDDP/SCRUM の併用	・QCD 意識(特に品質面) ・後戻り工数の削減 ・チームの一体感 ・継続的な改善意識 ・定期的なリリース	・不足要素なし ※XDDP と SCRUM の併用による負の作用は無い

表 8： 課題解決に有効な手法

課題項目	課題解決に有効な手法
１．後戻りに関する課題	XDDP
２．ソフトウェアを開発するチームに起因する課題	SCRUM
３．ソフトウェアのリリースタイミングに起因する課題	SCRUM

5.3 ソフトウェア開発を支える三要素についての考察

一般的に品質(Quality)・コスト(Cost)・納期(Delivery)を向上させるには，ソフトウェア開発を支える三つの要素である「プロセス」「人」「技術」の領域をバランス良く改善する必要があるとされている.[3]

本論文で提案した XDDP/SCRUM の適用プロジェクトにおいて,4 章の適用結果で示した通り,XDDP を適用することで「プロセス」領域に関する品質及び生産性に関して高い改善効果をあげることができた.一方，SCRUM を適用することでチームが成長し，「人」領域に関する改善効果をあげることができた.また，「プロセス」と「人」の領域の改善により，不要な後戻りや不要な人的問題に関する無駄を削減することができたことで，ドメインスキルやソフトウェアアーキテクチャの「技術」領域に時間を費やすことができた.

このように，ソフトウェア開発を支える三つの要素である「プロセス」「人」「技術」に関してバランス良く改善効果をあげることができた結果，品質(Quality)・コスト(Cost)・納期(Delivery)の向上につながったと推測する.

以上，三つの観点の考察からも，チームで派生開発を成功させるための手法として，本論文で提案する XDDP と SCRUM の適用は有効であるといえる.

6. 今後の課題

今後の課題を２点あげる.

１点目は，システムテストの重複である．今回のように開発が複数のイテレーションに分割した場合に，システムテストの重複項目が発生する．システムテストのテスト効率を上げるためにも，回帰テストには，積極的にテストの自動化を取り入れる必要がある.

２点目は，開発規模が大きくなった場合の SCRUM チームの構成である．今回は SCRUM チームに適しているといわれている１０名未満の構成であったため，SCRUM チームの構成による問題は無かったが，今後，１０名以上の人員で SCRUM チームを構成して開発を行う際には，SCRUM チームを複数に分けるなどの工夫が必要である.

7. おわりに

派生開発において迅速かつ最適の人員で高品質なソフトウェアのチーム開発を実現するために,派生開発に特化した開発プロセス的アプローチ(XDDP)と，ヒューマンマネージメント的アプローチ(SCRUM)の異なる二つのアプローチを適用して課題解決に取り組み，QCD 観点における改善効果とチームの成長を得ることができた.

XDDP のプロセスを実践することで，品質・コスト・納期に関して高い意識づけを持つことができた．また，SCRUM のプラクティスを実践することで，継続的な改善意識を持った自律型チームを形成することができた．その結果，高い QCD 意識を持った自律型チームに成長することができた．これが XDDP と SCRUM の併用の最大の効果だと考える.

参考文献

[1]清水吉男，『派生開発』を成功させるプロセス改善の技術と極意，技術評論社，2005
[2]Ken Schwaber and Jeff Sutherland, The Scrum Guide, Scrum.org, 2011
[3]独立行政法人 情報処理推進機構ソフトウェア・エンジニアリング・センター，プロセス改善ナビゲーションガイド，オーム社，2007