標準プロセスを肥大化させない補完型チケット駆動開発の提案

阪井　誠
SRA
sakai @ sra.co.jp

要旨

本論文では標準プロセスを肥大化させないプロジェクト運営のノウハウの蓄積方法として，補完型チケット駆動開発の利用を提案する．補完型チケット駆動開発では計画と異なるタスクがチケットとして記録されるので，チケットを参照することで，過去の想定外の事象に対してどのような対処を行ったか先人のノウハウを知ることができる．しかし，チケットからどのような情報が得られるか明らかになっていなかったことから，導入が容易であるにも関わらず広く普及しているとは言い難い．

そこで，補完型チケット駆動開発を実施したプロジェクトの追加タスクのチケットを分析した結果，７種類に分類が可能で，仕様変更，運用データ，準備作業など客先業務に固有の経験や，基盤の入れ替え，環境依存のテスト，追加の管理作業，障害の分析といった汎用的な経験が蓄積されており，標準プロセスに組み込まなくても類似プロジェクトの計画時やトラブル解決時のノウハウとして利用可能なことがわかった．

1. はじめに

ソフトウェア開発において組織の成熟は，ソフトウェアの品質や生産性の向上に重要である．日本においても古くから開発標準の制定や QC 活動などを通じて改善活動が行われてきた．特に 1990 年代中頃の CMM／CMMI ブームからは，標準プロセスとそれを通じた組織レベルの改善が行われてきた[1].

CMM／CMMI は広範な調査に基いており，より良い組織から集められた汎用的なベストプラクティスの集合でもある．このベストプラクティスを標準プロセスに組み込めば，ソフトウェア一般に生じる問題に対する解決策を標準プロセスに組み込むことが可能である．標準プロセスは組織内の標準として構築され，必要に応じてプロジェクトごとにテーラリングして適用される．すべてのプロジェクトは標準プロセスに基づいて実施されるので，各組織の顧客や文化に応じて標準プロセスを改良し，技術移転することで，継続的な改善を進めることが可能な仕組みである．特定の業務ドメインや顧客に特化した組織には効果的な方法である．

その一方で，ソフトウェアの適用分野は急速に多様化している．経験の少ないプロジェクトでは想定外の事象が発生してプロジェクトは混乱を招きやすい．このような想定外の事象の対処を目的として発生した作業は，実行時のダイナミックなプロセスの変更と言われ，ソフトウェア開発によくある事象としてソフトウェアプロセスモデリングのための例題として扱われている[2]．ダイナミックなプロセス変更の履歴は想定外の問題があったことと，その具体的な解決策を示す，いわゆるノウハウの源泉である．新しい業務ドメインなどで発生した想定外の事象のうち，頻度が高く影響の大きい問題は，問題の発生を防ぐ方策をなるべく前工程に組み込んでフロントローディングすると，より良いプロセスに改良できると考えられる（図 1 組織的な改善）．

このように想定外の事象が発生し，ダイナミックなプロセスの変更が行われたプロジェクトが終わると，本来ならその経験を今後に生かす目的で標準プロセスにそのノウハウを組み込んでいくことになる．しかし，このような過去に経験の少ないプロジェクトでは，以下のような問題があるので，組織レベルの改善活動では必ずしもうまく活用できない．

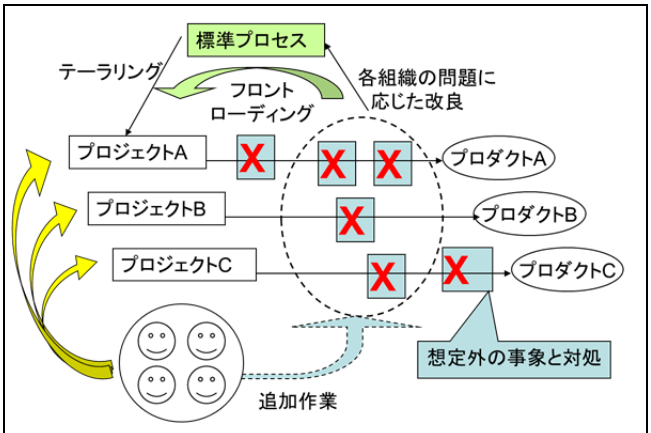

図 1 組織的な改善

- 新しい分野が増えるごとに情報を追加すると標準プロセスが肥大する
- 類似のプロジェクトが予想できない場合は標準化のモチベーションが高くなりにくい
- 経験の蓄積が必要なので，開発中の問題に対する支援ができない

特に標準プロセスの肥大化は，ウォータフォール型開発などの計画駆動型開発の問題点として，「最初にありとあらゆるものを包括した手法を開発し，（中略）たいていはかなりの額の浪費をする羽目になった」[3]と書かれており，従来から問題とされていた．

このような問題を解決する方法として，ダイナミックなプロセスの変更をすべて標準プロセスに組み込むのではなく，直接参照して活用することが考えられる．ダイナミックなプロセスの変更を収集した研究としては文献[4]がある．この文献では作業報告書から手作業でダイナミックな変更を抜き出して蓄積している．この研究ではノウハウの蓄積よりも標準プロセスの改善に注目しており，問題の原因に応じた対策を前工程で実施しようとしている．この方法は組織のプロセス改善の方法としては有効であるが，上に挙げた標準プロセスの問題を解決できるものではなかった．また，作業報告書から経験を抜き出す作業が必要なことから，標準化のモチベーションが高くなりにくい方法であった．

そこで，本研究では補完型チケット駆動開発で蓄積されるチケットに注目し，その可能性を検討した．チケット駆動開発とは，Redmine や Trac など ITS（課題管理システム）のチケットでタスクを管理する開発方法である[5]．全てのタスクを管理する完全型チケット駆動開発に対して，補完型チケット駆動開発は計画にない作業が発生した際にチケットが作成される．このため，補完型チケット駆動開発では，ダイナミックなプロセスの変更だけがチケット化されており，想定外の問題とその対策であるノウハウが自動的に記録されると考えた．

以下の章では，補完型チケット駆動開発を説明し，実際のプロジェクトのデータで検証したケーススタディについて述べ，その結果から，補完型チケット駆動開発によるノウハウの蓄積とその利用に関して考察する．

2. 補完型チケット駆動開発

本章では「チケット駆動開発」とチケット駆動開発の一種である「補完型チケット駆動開発」について述べる．

2.1. チケット駆動開発

チケット駆動開発は ITS のチケットを作業管理(タスクマネジメント)に用いる開発法である．障害票や課題票にあたるチケットを用いてタスクを管理することで，障害の管理と同じように，現状でどのようなタスクがあり，だれが担当し，どのような議論が行われ，どのような状態であるかをチケットで可視化する．またチケットには，今後のタスクの予定，作業中のタスクの状況，過去のタスク履歴を示しており，様々な条件で検索して利用可能である[5]．

チケット駆動開発は，たくさんの小さな修正を加えるシステムを開発されている中で，ITS の一つである Trac のチケット(障害票)を用いて開発プロセスを改善したことから始まった[6]．障害管理票であるチケットなしに，構成管理ツールにコミット(更新)してはいけない(No Ticket, No Commit!)というシンプルなルールで運用される．チケットによって情報共有されることでプロジェクトの柔軟性が高まるほか，膨大な作業はプロジェクトのリリース計画に従って構造化され，プロジェクト内で見える化される．また，作業やソースコードと関連付けられた多くの履歴が管理できるようになる．

このようなチケット駆動開発を導入すると，各担当者の日々の活動は，担当チケットの確認，作業実施，進捗更新という繰り返しのパターンになり，担当作業に集中することができる．また，リリースに向けても，リリース計画，チケット登録，チケット解決，リリース，ふりかえり，といった一連のプロセスが繰り返されるようになる．このような繰り返しの中で，プロジェクトにリズムが生まれ，繰り返しごとにプロセスが改善される[7]．

チケット駆動開発はこのように Trac ユーザから生まれたが，その後，Redmine ユーザにも広がり[8]，従来型の開発やアジャイル開発でも利用されている[9][10]．また，情報共有が容易でリアルタイムにメトリクスが収集可能なことから教育にも利用されている[11]．

2.2. 補完型チケット駆動開発

チケット駆動開発は，全てのタスクをチケット化する完全型チケット駆動開発と，WBS に基づく線表など従来型の管理で開発を進め，想定外の作業が発生したときに備忘録的にチケットを用いる補完型チケット駆動開発がある．

完全型チケット駆動開発は全てのタスクをチケット化するので，プロジェクトの進捗管理を一元化することができる．近年の ITS では標準あるいはプラグインでガントチャートをサポートしており，プロジェクトの管理にも用いること

が可能である. このようにすべてのタスクがチケット化され, チケットを中心にプロジェクトが運用される状況が本来のチケット駆動開発の姿であるが, 既存の標準プロセスに定められた管理方法を見直す必要があり, 導入は必ずしも容易ではない.

補完型チケット駆動開発は, 従来型の開発で想定外の問題が次第に明らかになる中で生まれた[5]. WBS に基づく線表など従来の管理手法で開発を進めるので, 標準プロセスを変更する必要がない. また, 想定外の作業が発生したときにのみ備忘録的にチケットを用いるので, 管理対象となるチケットの総数が少なくなるというメリットがある.

完全型チケット駆動開発はトップダウンに導入されることが多く, どちらかというとプロジェクト管理の方法となりがちである. これに対し, 補完型チケット駆動開発は社内的なプロジェクト管理から独立してチケットが運用されるので, 開発現場の情報共有や備忘録として活用される傾向がある[12].

組織的な改善と補完型チケット駆動開発の関係を図 2 補完型チケット駆動開発に示す. 組織的な改善方法であるフロントローディングの考え方では想定外の事象を生じさせた問題が起きないように標準プロセスを改良するのに対し, 補完型チケット駆動開発では, 想定外の事象の経験をチケットとして ITS に蓄積し, 類似プロジェクトの計画時や問題が発生した際に参照される. このような補完型チケット駆動開発のチケットは徐々に広がっていると思われるが, どのような情報が蓄積されており, どのように利用できる可能性があるかはあまり明らかにされていなかった. このような状況が, ITS すら使用しない前近代的なプロジェクト管理が残っている一つの要因であると考えられる.

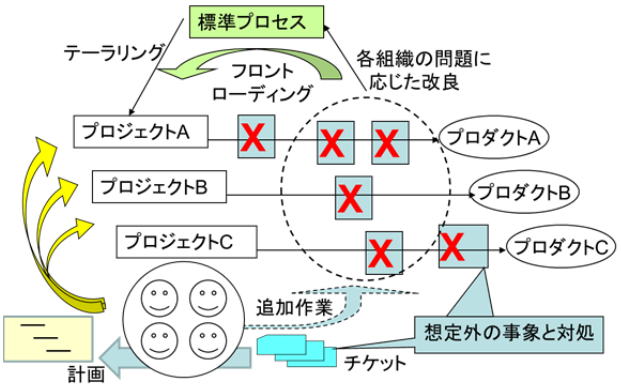

図 2 補完型チケット駆動開発

3. ケーススタディ

文献[12]に示す補完型チケット駆動開発を実施したプロジェクトのチケットを分類した. 文献[12]では補完型チケット駆動開発でプロジェクトがどのように活性化したかを報告しているが, 本研究ではチケットにどのようなノウハウが記録されているか検証することを目的とし, タスクチケットの内容に応じて分類した.

3.1. 対象プロジェクトの業務

検証の対象は Struts フレームワークをベースとする統合文教パッケージ UniVision のカスタマイズプロジェクトである. オープンなフレームワークを利用し, 高機能なソフトウェアを組み合わせることで, 少人数で様々な顧客に対応可能なシステムである. Struts フレームワークによってシステム内に同じようなコードが少なくなり, 個別の業務要件に対応する固有の開発をするだけで良いのである.

UniVision では, 対象業務ごとのサブシステムを複数組み合わせてシステムを構成できるほか, 個別要件にも柔軟に対応が可能である. オープンソースだけでなく商用 DB にも対応しており, 規模, 予算に合わせて, ハードウェアやミドルウェアの構成が選べるようになっている.

その一方で, カスタマイズ作業には固有のルールが多く, 複雑で難しい作業になりがちである. また, 少人数での部分的なカスタマイズを実施するだけで顧客専用の大規模なシステムが作れてしまうので, 工夫できる余地が少ない作業でもある. さらには, オープンソース実行環境の構成はバージョンアップによって日々変化するので, リリースまでに最新版を取り込んで動作を確認しないといけない. 一言で言うと, 複雑で大変な作業である.

3.2. プロジェクト概要

プロジェクトは最大で 8 人, 仕様変更の追加作業を含めて約 1 年間のプロジェクトであった. 全体の期間は長かったものの要件定義の期間が長い上に, 仕様がなかなか決まらず, 製造の期間が圧迫されていた.

プロジェクトメンバーのスキルは高いものの, 業務経験者が少なく, パッケージやミドルウェアの構成も UniVision としては初めて利用されるものがあるというリスクの高い開発であった. メンバーはプレッシャーの中で不安を感じ, プロジェクト全体に重苦しい雰囲気があった. 守りに入るメンバーもおり, コミュニケーションの悪いプロジェクトであった. また, プロジェクトへの参加時期や協力会社との関

係から経験の少ない者がサブリーダを担当していたことから，作業指示がうまくいかないこともあった．

システムテストの時期が近づいてくると，計画外の環境構築やリリース準備作業が必要であることが明らかになった．さらには環境に関連するバグまでが明らかになり，急激な作業の増加によってプロジェクトのコントロールが難しくなった．

このようにコミュニケーションが悪く，ゴールが見えない状況だったので，補完型チケット駆動開発を導入した．すでに ITS を障害管理に利用していたことから特に ITS の教育は行わず，「バグだけではなく，ソースを触るときや，WBS にない作業をするときは，チケットを登録してください！」と宣言して補完型チケット駆動開発を開始した．気づいた作業を備忘録的にチケットに記入し，共有することで，プロジェクト内のコミュニケーションが大きく改善した．作業量は少なくなかったが，全体の作業量が見えていたので効率的に作業を分担して何とかリリースすることができた．

3.3. 検証方法

上記プロジェクトのチケットを分類した．プロジェクトの計画時や問題発生時に参考にすることを想定して，障害チケットや未分類チケットは対象とせず，タスクチケットのみを対象として分類した．ITS のレポート機能でタスクチケットのみを抽出して一覧を作成した．分類にあたっては，類似しているチケットを順次まとめていく方法で分類した．

4. 分類結果と考察

4.1. チケットの分類結果

補完型チケット駆動開発を実施した対象プロジェクトの全 265 チケットのうち，計画以外の作業として追加されたタスク 77 チケットを分析した．チケットの内容を確認して，類似チケットごとに分類した結果， 客先業務に固有のチケット，汎用的な内容のチケットに大きく分けることができた．また，全体では7種類に分類が可能だった（表 1 チケットの分類結果）．表中にある各項目は以下の分類を示している．

客先業務に固有のチケット
- ・仕様：顧客の要望に基づく仕様変更
- ・データ：連携システムからの運用データ移行など
- ・準備：客先固有の資料作成などの準備作業

表 1 チケットの分類結果

客先業務固有のチケット			汎用的な内容のチケット			
仕様	データ	準備	基盤	テスト	管理	障害
11	25	6	17	12	3	3

汎用的な内容のチケット
- ・基盤：最新のシステムや DB などへの入れ替え
- ・環境：構築した環境依存のテストなど
- ・管理：リリース前の確認作業など追加の管理作業
- ・障害：障害に関連する分析作業

対象のプロジェクトは既存システムからの移行であり，外部システムとの連携もあったので，データに関するタスクが多かった．また，カスタマイズ元の統合文教パッケージ UniVision はオープンソースを多用したシステムであることから，パッケージやミドルウェアの更新などの基盤のタスクが多かったと考えられる．

4.2. 考察

補完型チケット駆動開発を実施したプロジェクトのチケットを分類した結果，チケットはプロジェクトの特性をあらわしており，客先業務に固有のチケットのほか，汎用的な内容のチケットが蓄積されていることがわかった．客先業務固有のチケットは，同一顧客の他システム構築時に参考になると考えられ，汎用的な内容のチケットは他の顧客に同一システムを導入する際やトラブル発生時に参考になると思われる．もし，同一システムの導入が継続するのであれば，標準プロセスにチケットの内容を一般化して組み入れると良いと考えられる．

蓄積されたチケットは，そのタイトルを見るだけでもどういった作業が必要になるかなど経験の少ない管理者・開発者には大いに参考になると思われた．その反面，技術的な TIPS は詳細情報を読まないとわからないものがあった．具体的には，デプロイ時のサービス停止時間を最短にする目的で，あらかじめ war ファイルを配置しておき，予定時刻になってから TOMCAT の再起動を実施していた．このような TIPS は仕組みを知っていれば明らかであり，詳しくは書かれていなかったと考えられる．社内教育や OJT による技術移転が必要と考えられる．

ダイナミックなプロセスの変更として分類結果を見ると，追加タスクの収集はできたが，削除や修正に関しては収集できなかった．これは，補完型チケット駆動開発は，既存の管理方法に基づくので，元々のタスクの削除は管理されないからである．また，元々のタスクを変更した際に

作業が詳細化されればチケット化される可能性はあるが，基本的には記録されないと考えられる．これらは今後のプロセスのノウハウあるいはバッドノウハウとして，標準プロセスで支援する必要がある．

もし，このプロジェクトを完全型チケット駆動開発で実施していたなら，ベースとなるプロセスがチケット化されているので，その削除や修正が記録できる可能性がある．その反面，チケットはWBSよりも粒度が細かいことが多いので，安定したプロジェクトでないと差分がとりにくい．今回のプロジェクトのように仕様が確定せずに段階的に確定する状況では，常にチケットが更新されてしまい，ベースとなるチケットと差分のチケットが区別できないからである．その点では，削除と変更の情報は得られないものの，補完型チケット駆動開発の方がノウハウの収集が容易であるといえる．

補完型チケット駆動開発のチケットは数も少なく，容易に検索ができるので，過去のプロジェクトを参考にして，リスク見積，計画，問題が発生した際の対策，などの参考にすることが可能である．プロジェクトのノウハウを個人に閉じることなく，類似プロジェクトや将来の保守の際に効果を発揮すると考えられる．

今回は補完型チケット駆動開発をシステムテスト以降で実施したが，より早いフェーズから実施した場合や，規模や対象業務が異なる場合には，チケットの比率が異なる可能性がある．しかし，プロジェクト毎に固有の問題は存在すると考えられるので，一定の効果は期待できると考えられる．

5. まとめ

標準プロセスを肥大化させないプロジェクト運営ノウハウの蓄積方法の一つとして，補完型チケット駆動開発の利用を提案した．補完型チケット駆動開発を実施したプロジェクトの追加タスクのチケットを分析した結果， 7 種類に分類が可能で，仕様変更，運用データ，準備作業など客先業務に固有の経験や，基盤の入れ替え，環境依存のテスト，追加の管理作業，障害の分析といった汎用的な経験が蓄積されており，類似プロジェクトの計画時の作業項目漏れの確認や，トラブル解決時にキーワードで類似障害を検索して対応策を検討するといった利用方法が可能なことがわかった．

今回のチケットの内容のうち，汎用的な経験は考慮漏れのチェックリストなどに一般化して標準プロセスの改良にも用いることができる．しかし，発生頻度の低い問題の対策を無秩序に追加していくと，標準プロセスはどんどん肥大化して利用が困難になってしまう．そこで，補完型チケット駆動開発を利用して必要な時に類似プロジェクトを検索すれば，実際に起きたことを具体的に確認でき，参考にすることが可能である．

プロジェクトには予見できないことが数多く存在している．Humphrey はこれを要求の不安定さと呼び，使わないとわからない未知の要求，細目が流動的な不安定な要求，実現方法の詳細が理解されていない誤解された要求，の 3 つを挙げている[14]．今回のチケット分類を要求の不安定さにあてはめると，それぞれ仕様変更，運用データ，準備作業に相当する．Humphrey は要求の不安定さの解決策としてプロトタイピングを薦めているが，Humphrey 自身が問題点を挙げているように，プロトタイピングには正解がなく，そのコントロールは難しい．

補完型チケット駆動開発で得られるタスクは標準プロセスに追加されたタスクであり，そのプロジェクトに特有のノウハウである．経験を蓄積する目的でプロトタイピングを実施しなくても，チケットからノウハウを得ることで発生しうる事象の予想がある程度可能である．また，想定外の事象が発生した際の対策の検討にも有効で，プロジェクトの運営を堅牢にするものである．しかし，効率化などを目的に削除や修正が行われた作業の記録は含まれておらず，また汎用的な TIPS は記録されにくい．標準プロセスの改良や社内教育などと組み合わせて利用すると良いと思われる．

チケット駆動開発を実施するには ITS などのチケットを管理するツールが必要になる．しかし，近年のトレンドとして知っていても，従来の標準プロセスが制約になって実施できない組織が未だに多い．このため，標準プロセスに取り込まれなかった多くのノウハウが，個人の経験として蓄積されてしまい，多くのノウハウはプロジェクトが終わるたびに忘れ去られている，今回の検証のようにチケットには多くのノウハウが含まれており，このような検証を通じて補完型チケット駆動開発を普及させていきたい．

参考文献

[1] フュジェッタ，ウルフ，“ソフトウェアプロセスのトレンド”, pp.212-213, 海文堂, 1997.
[2] 井上，松本，飯田，“ソフトウェアプロセス”, pp.209-221, 共立出版,2000.
[3] ベーム，ターナー，アジャイルと規律，p.189，日経 BP 社, 2004.
[4] Sakai, Matsumoto, Torii, “A new framework for improving software development process on small

computer systems,” International Journal of Software Engineering and Knowledge Engineering, vol.7, no.2, pp.171-184, 1997.
[5] 小川，阪井，“チケット駆動開発”，翔泳社, 2012.
[6] まちゅ，“チケット駆動開発 … ITpro Challenge のライトニングトーク (4) –”，まちゅダイアリー，http://www.machu.jp/diary/20070907.html, 2007.
[7] 小川，阪井,「チケット駆動開発- BTS で Extreme Programming を改善する-」，ソフトウェア品質シンポジウム 2009,
http://www.juse.or.jp/software/83/attachs/ippan_2-1.pdf
[8] 小川，阪井，“Redmine よるタスクマネジメント実践技法”，翔泳社, 2010.
[9] 岡，三宅，“本当に使える開発プロセス”，日経 BP 社 2012.
[10] 前川，西河 誠，細谷，“わかりやすいアジャイル開発の教科書”，ソフトバンククリエイティブ, 2013.
[11] Igaki, Fukuyasu, Saiki, Matsumoto, Kusumoto, "Quantitative Assessment with Using Ticket Driven Development for Teaching Scrum Framework," In Proceedings of the 2014 International Conference on Software Engineering(ICSE2014), 2014.
[12] 阪井，“チケット駆動開発によるプロセス改善 現場重視，管理重視，それとも情報共有重視”，SPI Japan 2013, SPI コンソーシアム, 2013.
[13] 阪井，“チケット駆動開発によるプロジェクトの活性化 ―見える化と運用ポリシーがプロジェクトを変えた”, SPI Japan 2010, SPI コンソーシアム, 2010.
[14] Humphrey，“ソフトウェアプロセス成熟度の改善”，pp.276-295，日科技連, 1991.