TiDD：チケット駆動によるアジャイル開発法

株式会社 SRA 先端技術研究所
阪井　誠

概要

本報告では，アジャイルソフトウェア開発に生じる問題と，近年，注目されつつあるチケット駆動開発(TiDD)について述べる．TiDD は Redmine や Trac のような障害管理ツールを中心に開発ツールを駆使して，アジャイル開発をマネージメントする方法である．TiDD では，障害管理ツールの障害票に相当するチケットを，アジャイル開発のタスク管理に利用する．また，チケットと連携した構成管理ツールによってリリース後のコードと開発中のコードの同期をとる．また, コード修正時にはビルドツールやテストツールによる自動化によって品質を維持するのである．

１．はじめに

XP(Extreme Programming)[1]をはじめとするアジャイルソフトウェア開発(以下，アジャイル開発と呼ぶ)は，変化する顧客の要望を取り入れることが可能であることから，普及が進んでいる．アジャイル開発はそのプラクティスによって，さまざまな効果が得られる[3]．しかしその反面，アジャイルソフトウェア開発には固有の難しさがある．

アジャイルソフトウェア開発は以下のような特徴と困難な点がある．

(1) 短期間での繰り返し開発

アジャイルソフトウェア開発では「Embrace Change (変化を抱擁せよ)」[1]と言われ，開発中に明確になっていく顧客の要望を順次取り入れる．従来の表現を用いるならばスパイラルモデルでの開発，あるいは常に進化的保守が行われている状態と考えることができる．

開発の期間が非常に短い小さな開発が繰り返されるので，作業の管理が困難である．XP ではタスクカードを掲示板に貼ることによって進捗を見える化するが，全体をどのようなイテレーションに分割するかというロードマップや，保留となった障害のようにイテレーションをまたがるタスクの管理，日々の個人のタスクの状況など，管理が難しい面があった．

(2) 開発と評価の同時進行

アジャイル開発では繰り返し開発が行われる中で，ユーザに対して複数回のリリースが行われる．これは，正式リリースのほか，ユーザによる評価を目的として行われる．ユーザにリリースした後も継続してソフトウェア開発が行われるので，ソース管

理の観点でみると並行開発，あるいは，プロダクトラインと同様の開発形態になっている．

このように複数のソースコードが存在する状態で何らかの障害が発生すると，その管理が難しくなる．ひとつの障害対策を 2 か所で実施するためには，抜けが生じないような手順の確立が必要であるほか，2 か所の仕様や実装が異なる場合には個別の対策を取らないといけない．また，個別に対策を実施した場合，将来それらのコードを統合が必要となることもある．

(3) 頻繁なリファクタリング

アジャイル開発ではソースコードを重視した開発が行われる．同じ機能を維持しながら，より良いコードに変更するリファクタリングを常に行うことで，保守性を維持するのである．

このような積極的なコード修正は，すでに動作しているプログラムが動かなくなるのではないかという品質面の不安が生じる．そこで，修正時にテストの抜けを生じさせないように，自動化ツールを用いたテストが必要となる．

２．チケット駆動開発

TiDD は Redmine や Trac のような障害管理ツールなど開発ツールを駆使して，アジャイル開発で生じる問題を解決する．TiDD では，障害管理ツールの障害票に相当するチケットを用いて，多数のタスクの管理，複雑な構成管理，リファクタリング時の品質の維持，といった開発に関わる作業のワークフローを管理する．

従来，障害管理ツールを利用する際に，要望も障害も同様に扱うという Issue tracking が行われていた．TiDD では，これをソフトウェア開発全般に拡張するものである．特にソースコードの変更はチケットと関連付けなければ認めないというルールを定めて設計・実装・試験をチケットで管理する．障害管理ツール上にはチケット集計機能があるので，それを用いて進捗の管理をする[4].

このチケットの利用によってアジャイル開発でのロードマップや多数のタスク，個人の状況などが見える化され，全メンバー間で情報共有できる[5]．また，チケットと連携した構成管理ツールによってリリース後のコードと開発中のコードの同期をとることで，複雑な構成管理が容易になる．また，コード修正時には，ビルドツールやテストツールによる自動化でリファクタリング時の品質が維持されるのである[6].

TiDD は新規の開発方法であると共に，既存の開発方法の集大成でもある．修正作業のチケットによる管理は従来から行われており，それを拡張したと考えられる．また，障害管理と構成管理の関連付けは，多くの企業で行われていた．また，品質を保証する目的でテストやビルドの自動化は古くからおこなわれていた．だが，それらは個別に行われており，一連の作業の連携はあまり明確でなかった．

しかし，ソフトウェアを取り巻く環境の変化からアジャイル開発が行われる中で，多くの問題がこれらの方法の連携を促した．アジャイル開発はライトウェイトプロセスと言われるものの，その実施は簡単ではない．従来の開発法と比べると，スキルの高

いメンバーが必要な割合が高いと言われている[2]．ソフトウェア開発に対する高度な技術のほか，従来に比べて複雑な手順の作業を必要としているのである．TiDD で行われる様な，アジャイル開発に使われる多くのツールの連携は，必然だと言える．

なお，TiDD では方法論に合わせてツールを作るのではなく，既存のツールを使う中で方法論が作られてきた．このため，従来の方法論が必要としたツール開発のコストは基本的にかからないという特徴がある．

３．TiDD の現状

TiDD はすでに開発現場で実施されている．TiDD を利用することにより，アジャイル開発の効率化のほか，問題点の見える化により開発者と協力してプロジェクト開発のマネージメントが行える事が報告されている[5]．

文献[5]では Redmine を用いて TiDD を実施している．Redmine のロードマップ機能でリリース予定のバージョンを指定することで，イテレーションの計画と管理が行われている．また，イテレーション作業をタスクに分割してチケットに対応付けることで，作業のワークフローの自動化と進捗管理を行っている．

TiDD によるアジャイル開発では，遅延タスクや進捗が明らかになったほか，チケットの取捨選択によって計画ゲーム(リリースの範囲の決定)がツール上で行えるようになった．これらのメリットはチケットを随時更新することで得られるが，ツール連携による効果から開発者も協力的であったとされている．

４．おわりに

本報告では，障害管理ツールをはじめとするツールを用いたチケット駆動開発(TiDD)について，その背景と定義について述べた．タスクをチケットに割り当てることで，作業のワークフローが管理され，ツール連携による自動化でソフトウェアの信頼性向上を図っている．

TiDD はアジャイル開発のツール化を進めた際に，自然発生的に生まれてきた．従来のように方法論に合わせてツールを作るのではなく，ツールの利用中に生まれたアイデアから方法論を作っているので，現時点ではツール開発のコストがかからない方法論である．

日本の環境ではアジャイル開発に適したプロジェクトルームを用意することは難しいが，TiDD ではタスクカードを貼るための掲示板が必要なく，通常のオフィスで実施可能である．また，開発者も協力的であるとの報告もあり，今後の普及が期待される．

日本 XP ユーザーグループ関西(XPJUG 関西)ではオープンな議論が行われており，ツールの連携や具体的な実施方法をプラクティスとしてまとめ，より普及を図っていく予定である．

参考文献

[1] K. Beck, C. Andres, Extreme Programming Explained: Embrace Change (Xp), Addison-Wesley, 2004.

[2] B. Boehm, R. Turner, アジャイルと規律, pp.66-69, 日経 BP 社, 2004.

[3] O. Kobayashi, M. Kawabata, M. Sakai, E. Parkinson, Analysis of the interaction between practices for introducing XP effectively, Proc. ICSE'06, pp.544-550, 2006.

[4] 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#p01, 2007.

[5] あきぴー, Redmine でチケット駆動開発を実践する～チケットに分割して統治せよ, http://www.slideshare.net/akipii.oga/redmine-presentation?type=powerpoint, 関西オープンフォーラム(KOF2008), 2008.

[6] 中西庸文, ゼロ機能リリースのもうひとつの側面 ～ ワーキングスタイルを変える開発基盤をまず構築しよう ～, XP 祭り関西 2009, 2009.

謝辞

TiDD の名付け親である，えと～さんはじめ，XPJUG 関西の皆さんに感謝します.