
## 質問

Q: ソフトウェア開発が楽しいですか？

A: ものづくりだから楽しいはず
- しかし、不完全な人間が、間違いを犯さない計算機と戦わないといけない
- ミスをしてはいけないというプレッシャーを感じることも多い

★TiDDはそんなプロジェクトを元気にしました！

## チケット駆動開発（TiDD）とは

- BTS（ITS)を使う
- チケットを用いて作業の管理をする
- チケットがなければ（構成管理上の）更新をしない
- アジャイル的(XPの4つの価値)
  - シンプルな管理、見える化によるコミュニケーション改善、未完了作業のフィードバック、実施する勇気が必要

⇒ 様々な実施方法がある

## TiDDの形態

- 完全チケット方式(Redmineの本ほか)
  - チケットですべての作業を管理する
  - 管理が集約される
  - プロセスを変更するので社内調整が必要
- 補完チケット方式（今回の発表）
  - 既存の管理は変更しない
  - 計画外の作業をチケットで管理する
  - こっそり開始できる(でも報告は必要<=後述)

## チケットの管理方法

- ワークフロー型
  - 起票や終了には権限が必要
  - 仕様の一貫性を保障できる
- オープン型
  - 誰でも起票・終了できる
  - 機敏さ、自由がある

## TiDDの導入範囲

- 工程
  - 全工程、一部の工程
- プロダクト
  - 全体、一部
- メンバー
  - 全員、一部

## どのように導入するか

- 全体に適用
  - –プロセスがシンプルになるが、チケットが増える
- 部分的に導入
  - –細かなチケットが作成可能、プロセスの強化
  - –作業漏れ防止、コミュニケーション、などに効果

⇒ どちらが良いかではなく、ふさわしい方法がある

## 目次

- TiDDとその分類（説明ずみ）
- TiDDの経験談
  - –プロジェクト概要
  - –導入時の工夫
  - –結果、効果
  - –注意点
- まとめと感想

## TiDDの経験談（概要）

- 文教パッケージのカスタマイズ(最大8人x1年)
  - 短納期 & 仕様の決定遅れ・変更
  - スキルは高いが経験者が少ない(リーダは途中交代)
  - 初めての組み合わせ（サブシステム、ミドルのバージョン）
  - 義務感と不安、重苦しい雰囲気、閉塞感、、、
  - 守りに入るので、コミュニケーションが悪い

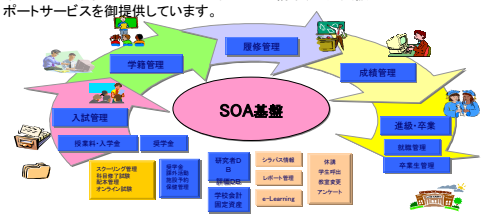
システムテストの時期になると、計画外の環境構築やリリース準備作業が明らかになった（環境に関連するバグも、、、）

⇒ そうだ！チケット駆動開発をしよう！

## 総合文教ソリューション UniVision

**UniVision**は入試管理、学籍管理、教務管理といった基幹事務機能を中心に学生サービスに関する情報全般を統合的に管理する統合文教ソリューションです。

また、2009年度より、新たに**法人系システム（学校会計、固定資産）**もサービス開始しております。システムコンサルティングから構築、運用支援にいたるトータルサポートサービスを御提供しています。

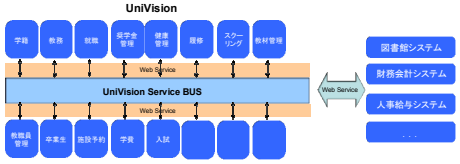



【UniVision全体概念図】

10

## 総合文教ソリューションUniVisionの特徴

- Strutsベースのパッケージ
  - 複数のサブシステムを組み合わせて構成できます
  - 個別要件にも柔軟に対応します
- オープンシステムにも対応
  - お客様の規模、ご予算に合わせて、ハードウェアやミドルウェアの構成が選べます






## オープンなフレームワークだから、、、

**長所**
- 同じようなコードが減る
- 業務要件に固有のコードだけで開発が完了

**短所**
- お約束が多い複雑な作業で、気が抜けない
- 工夫できる余地が少ない
- 少人数で大規模なシステムが作れてしまう
- 実行環境の構成は日々変化

=> ややこしくて、**しんどい！**

## いざ！導入

- 補完チケット方式
  - WBSと併用(と言っても更新作業は全てチケットあり)
- オープン型
  - だれでもチケットが作成できる
- システムテスト以降
- システム全体
- メンバー全員
- trac(単独)

・・・ SRA共通開発環境(trac,subversion,mailmanの仮想環境)

## はじめの一歩

- 宣言と実行
  - バグだけじゃなく、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！
- 環境の準備
  - レポート(チケットの一覧)の作成
    - bugのみ、taskのみ、その他、など
    - 抽出条件(WHERE句)や表示項目(前に"_"があると表示されない)は自由に変更できる
  - 権限の追加
    - tracの権限の設定は堅いので、チケットを修正できるようにmemberにTICKET_ADMINの権限を与えた
      (リスクを考慮して設定してください)

## 結果

- チケットの数
  - システムテスト：31
    - データの準備、環境準備、BUG関連で増えた作業、細かな仕様変更など、手順書にない1回だけの作業
  - 本番環境構築：42
- 作業漏れ減少！納期までに作業が完了！

(知らないこと、気付かないことはできませんでした、、、orz)

それ以外にも、メンバーに変化が、、、

## 目が輝いた！

サブリーダクラスなのに遠慮をしていたメンバーが、**生き生き**しだした

- 「チケットを切ってもいいですか？」
  ⇒ 義務的な作業からの解放
- 「チケットを切っておかないと忘れてしまう！」
  ⇒ すくに使いこなしていた
- 「ちゃんとクローズしてね」
  ⇒ 他の人に指導をしていた！
- 「残っているチケットが多くてわかりにくいから整理しますね」
  ⇒ 今後のことも考えている

## しまった！

- 上司への説明をしていなかった
  - 通知メールが多いので、不具合が多いのかと心配された

## 注意すべき点

- チケットの登録忘れ
  - チケットがルーチンワークになっていないと忘れがち
- チケットの実施忘れ
  - 他のチケットがないときにチェックを忘れる

⇒ 粒度の大きいチケットの後で発生
⇒ 粒度が小さいとリズムが生まれて、発生しにくい
(リーダが管理すれば防げますが、自主性が大事)

- 上司への連絡　(^_^;

## TiDDの効果(XP的な表現)

- 変化を抱擁できる
  - 当初想定しなかった作業を管理できた
- コミュニケーション
  - 見える化され全員が状況を把握
  - 他の人のチケットも自由に登録
    - それまではサブシステム間の連絡はリーダが担当
  - お互いに相談・協力するようになった

## TiDDの効果(スクラムの価値で)

- コミットすること
  - 自主的にチケットを発行し、実施した
- 集中すること
  - 担当チケットの作業に集中できた
- オープンであること
  - 通知メールやレポートで、プロジェクト全体を見渡せた
- 敬意を払うこと
  - 守りに入らず、助け合うようになっていた
- 勇気を出すこと
  - 前向きになる勇気が得られた！

K. Schwaber, M. Beedle, アジャイルソフトウェア開発スクラム, ピアソン・エデュケーション, pp.165-174.

## まとめと感想

- システムテスト以降にTiDDを導入
- 計画できていなかった作業が管理できた
- プロジェクトが元気になった

- 感想
  - 意外と簡単！(備忘録のつもりで)
  - もう少し上流でも使ってみたい(内乱だけでなく外乱も扱う)
  - Redmineが便利そう
  - 大学のH君を思い出した

## 大学の友人H

- 切符をどこに入れたのかを忘れてしまうからと

「ズボンのポケットに入れたから覚えといて！」

⇒「それぐらい覚えておけよ！」と内心思っていた

今から思うと
- 何かのトラウマがあったのかもしれない
- 本当に覚えるのが、苦手なのかもしれない
- 人に話すと覚えるという記憶法なのかもしれない

実はチケット駆動開発に似ている

## みんなで協力すれば楽しい！

- 忘れそうなことをみんなと共有することで
  - 緊張から開放されて安心できた
  - 忘れていたときは、教えてもらえた
  - もしものときは助け合った
  - トラブルは起きなかった

「楽しいし、元気が出た」

★ソフトウェア開発が、そういう仕事であってほしいです。
ぜひ、みなさんもTiDDを実践してください！

## 元気が出るチケット駆動開発
### - 補完型TiDDの経験 -

おわり