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-

株式会社SRA
阪井　誠

チケット駆動開発(TiDD)

- ITS(BTS)のチケットを障害、課題だけでなく、個人とプロジェクトのタスクを管理する
- 構成管理、Wiki、継続的統合などツールをチケットに連携させて自動化する
- 構成管理などのプロジェクトの情報をチケットに関連付けて、コミュニケーションを支援する

⇒ 現場から始まった改善活動

チケット駆動開発の効果

ツールを有効活用してプロジェクトの負担を減らす

- 過去:
  - 履歴の活用(経緯の確認、ノウハウの利用)
- 現在:
  - 障害、課題、タスク、ツール実行の管理
  - 情報共有、自動化、コミュニケーション
- 未来:
  - 計画、備忘録、リスクの見える化

チケット駆動開発の課題

多くの利用実績があるものの、事例があまり公開されていないので、以下のような問題がある

- どのような利用方法があるかわからないので、常に工夫が必要とされる
- 導入しても効果が出るとは限らない
- 組織的な改善に至るまでの道のりが長い

=> より多くの事例を分類して報告する

チケット駆動開発の事例

以下の3種類・4分類の事例を報告する

- 現場重視のプロジェクト
  - 備忘録的に利用して作業漏れを防止（未来）
- 情報共有重視のプロジェクト
  - 履歴活用（過去）と段階的開発の管理に利用（現在）
- 管理重視のプロジェクト
  - トレーサビリティの管理（現在）
  - オフショアでの課題管理、QA（現在）

現場重視のプロジェクト

- 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)
  - オープンフレームワークの初めての組み合わせ（サブシステム、ミドルウェアのバージョン）
  - 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、、、）

⇒　気付きの共有にチケット駆動開発を利用

チケット駆動開発の導入

- 宣言と実行
  「 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！」

- 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
  - 権限の追加
    - tracの権限の設定が堅かったので、チケットを修正できるように
      全ての権限を与えた(リスクを考慮して設定してください)
- 教育
  - なし(パッケージチームとのQ&Aでtracの利用経験があった)

現場重視のプロジェクトの結果

- システムテスト以降にTiDDを導入
- 備忘録のつもりで気づいたことをチケット化した
- 手順は順次Wikiにまとめた
- 計画外の作業を管理できた
- 作業を見える化することで
  コミュニケーションが向上した
- メンバーが前向きになり、
  プロジェクトが元気になった

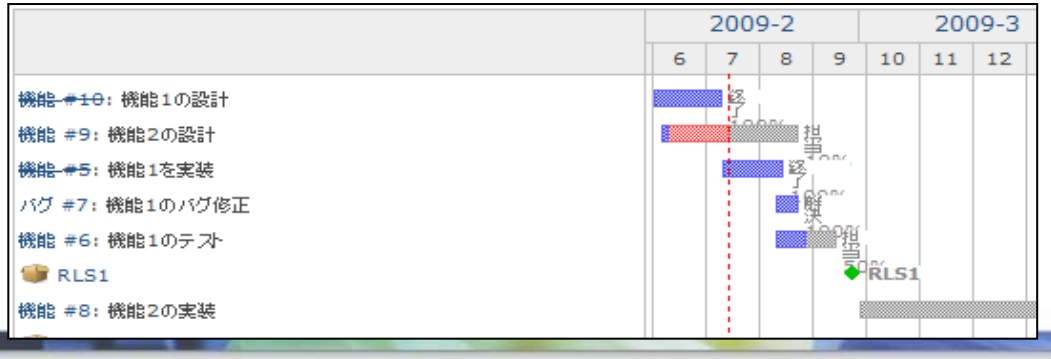
＊阪井, “チケット駆動開発によるプロジェクトの活性化
―見える化と運用ポリシーがプロジェクトを変えた”,
SPI Japan 2010, SPIコンソーシアム, 2010.

情報共有重視のプロジェクト

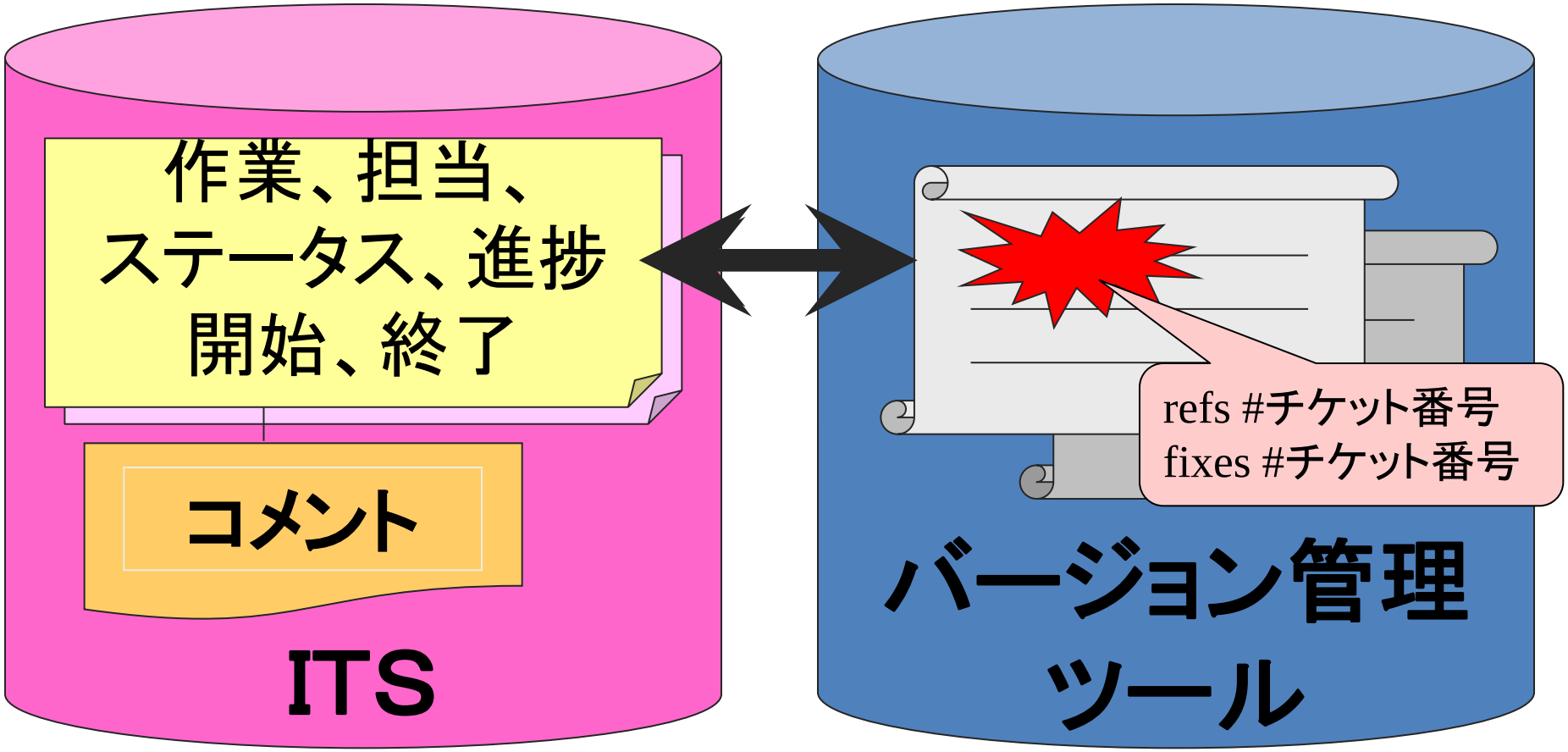
履歴を重視。課題管理や進捗管理をチケット化

- No Ticket, No Commitを実践して履歴を活用
  - チケットのない作業は許さない
  - 過去の履歴を検索して活用
- 障害だけでなく、課題管理や進捗報告にも利用
  - Q&A・課題はチケットで管理
  - ガントチャートも利用

次頁参照



ツール連携


作業、担当、
ステータス、進捗
開始、終了
コメント
ITS
refs #チケット番号
fixes #チケット番号
バージョン管理
ツール

情報共有重視プロジェクト結果

過去の履歴を重視するとともに、課題管理や進捗管理をオンライン化した

- 負担が少なかった
  - 経験者が実施していた
  - 現場の自主性を重視した
- 効果もそれなりにあった
  - 拠点の異なるステークフォルダと連絡が容易
  - 情報がチケットに一元化でき、朝会も効率化
  - 過去の経緯を容易に知ることができた

管理重視のプロジェクト

管理の支援に利用した

1）トレーサビリティの管理

- 要件と作業を関連付けた

2）オフショアでの課題管理、Q&A

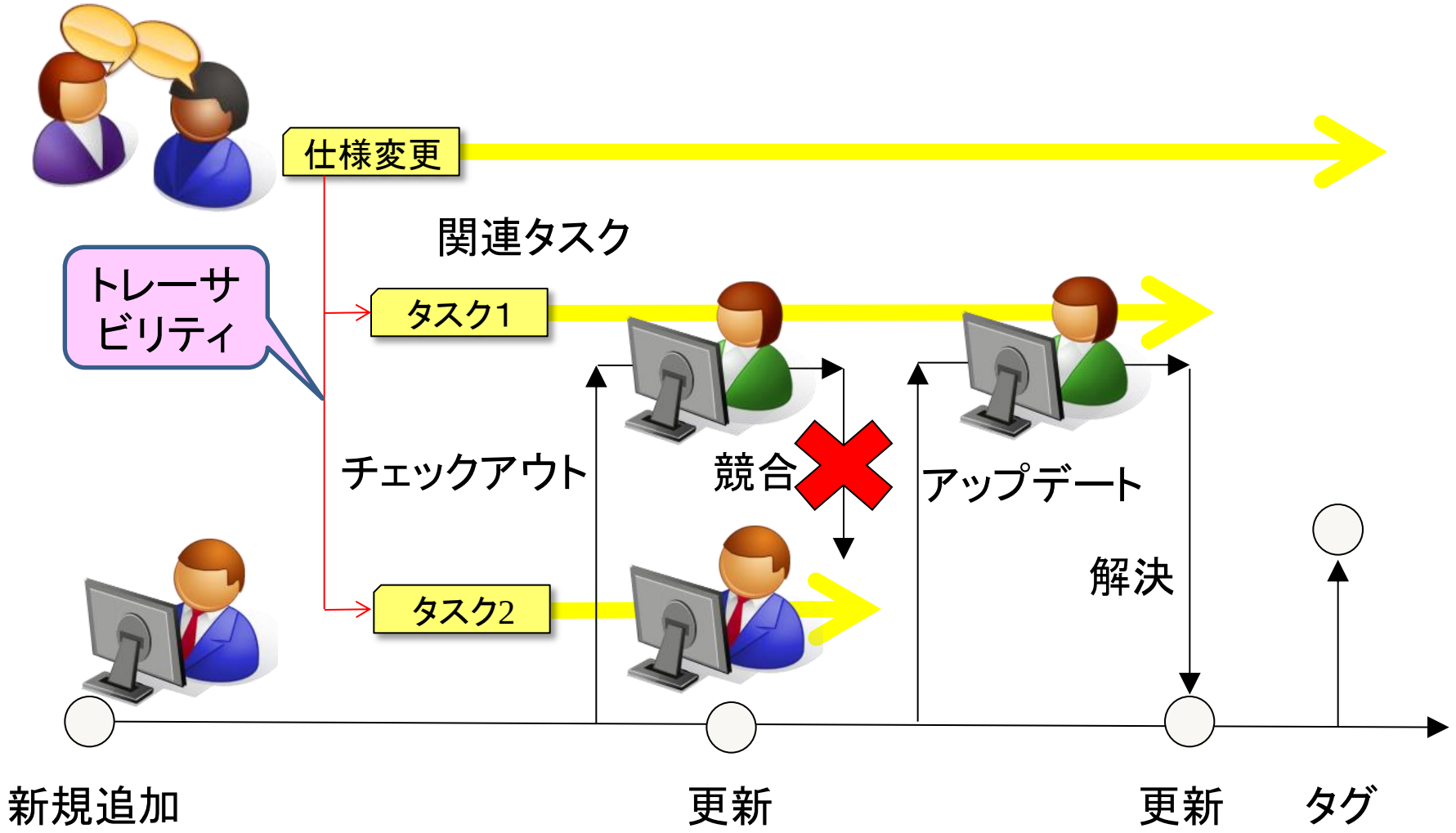
- Q&A・課題はチケットで管理
- ワークフローを定義

管理重視1:トレーサビリティ

- 要件と作業の紐付けを管理が必要
  - チケットの種類を要件とタスクとした
  - 相互に関連付けた
- 一覧の作成
  - 順序をタイトル・属性で持たせた

次頁参照

バージョン管理とチケット（仕様変更）



管理重視1:トレーサビリティの結果

- 要件と作業の紐付けを管理
  - 線表(ガントチャート)と併用していたので要件のチケットは管理のみに用いた
- 一覧は仕様変更時の対応が大変
  - タイトルや属性でソートしていたので変更が大変
  - 関連の変更も難しかった
- あらかじめ利用方法を検討すればよかった
  - Redmineなど親子関係が標準のITSの利用
  - エクセル連携など変更方法の工夫

経験を踏まえた

管理重視2:オフショアでの利用

- コミュニケーションが問題になりやすい
  - 状況が見えない
  - 履歴の検索・確認が必要
  - ボトルネックが発生しやすい
- 複数拠点
  - 通信プロトコルに制約(ファイル共有不可)
  - メールは使いにくい

メールではうまくいかない理由

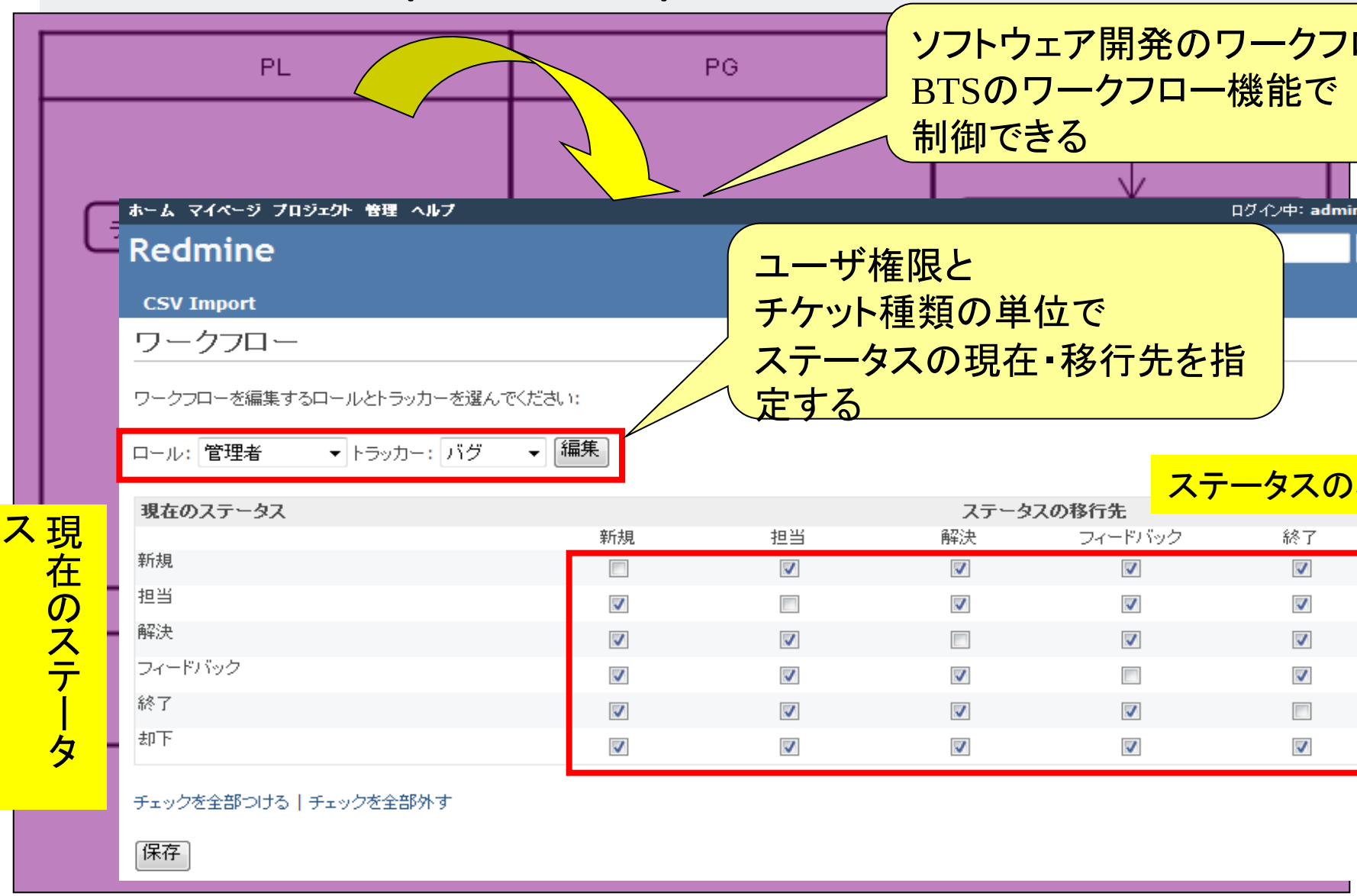
- 情報の保管場所
  - 個人の管理（本人以外は検索できない）
  - MLサーバーで管理（分類がないので検索が困難）
  - 閉鎖的になり、主要な関係者以外は見ることができない
- 情報の関連付け
  - ソースコードと（更新と理由）の関連付けができない
  - Wikiなどにまとめても、関連付けられない
- 情報の内容
  - 承認のワークフローがなく、ステータスがわからない
  - 検索のキーがないので探しにくい

ワークフローの検討

- メールを原則禁止、チケットを利用
  - 質疑応答や議論の経緯を残す
  - カスタムクエリで状況を容易に確認できるようにした
- 課題チケットは発注側の責任者が閉じる
  - ボトルネックになるが、最終確認ができる
  - チケットの種類を工夫して作業を効率化
- 無駄に厳密にしない
  - ボールの持ち主は、ステータスでなく担当者
  - 担当がいなくても情報共有、更新ができる

次頁参照

ワークフロー設定(例：Redmine)



SQiP2009発表資料より ©小川明彦, 阪井誠

管理重視2:オフショアでの利用の結果

- メールとファイル共有による開発が一変
  - 情報共有が容易になった
  - 状況の確認と履歴の検索が容易になった
  - ファイル共有によるロックの問題が解消
  - 安全に修正作業ができる

=>「もっと早く使っておけばよかった！」×３

まとめ(1/2)

	現場重視	情報共有重視	管理重視	
目的	作業漏れ防止	段階的計画、保守性向上	1) トレーサビリティの確保	2) オフショア開発の管理
チケット利用法	備忘録・情報共有	進捗、履歴	トレーサビリティ・進捗	課題管理、QA
プロジェクト数	2	3	2	1
チケットの経験	障害管理のみ	あり	あり	一部
モチベーション向上	大	中	小	大
管理面の効果	中	中	大	大
コミュニケーション向上	大	中	中	大
強制	なし	なし	あり	あり
負担感	小	なし	大	小
結果	作業量が明確になり、モチベーションが向上	管理が容易。必要十分な記録を自主的に実施	効果あり。要件変更時の負担が大きかった	ファイル共有とメールの混乱が解消
感想	危機感と目的意識があり、有効に使えた	報告も簡素化され、作業が進めやすくなった	面倒。負担削減の方法があったかもしれない	もっと早く導入すればよかった

まとめ（2/2）

- チケット駆動開発（TiDD）には様々な実践方法がある
  - 本報告は3種類・4分類を示した
  - より多くの知見を集め、多様な問題の解決を容易にして快適なプロジェクトを実現したい
- TiDDは現場作業の改善の中で生まれ、普及した
  - 現場の作業を考慮し、意見をうまく取り入れることがより良い改善の秘訣
- 「その最大限の能力を最大限発揮できるようメンバーを動機付け、コーチし、後押しする」
  - W・ハンフリー「TSP ガイドブック：リーダー編」翔泳社, 2007 .

おわり

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 -