ソフトウェア工学

(5-6) アジャイルソフトウェア開発

国立情報学研究所　　石川　冬樹

f-ishikawa@nii.ac.jp

http://research.nii.ac.jp/f-ishikawa/

目次

- アジャイルソフトウェア開発概要
- アジャイルプラクティス

「古典的な」アプローチに対する反省

- 計画指向・定型化指向に偏りすぎた
  - 十分達成可能で有用な計画を事前に立て，基本的にそれに従うという方針であり，変化・適応をあまり想定していない
  - コードが得られ，価値を得たり確認したいできるまでの時間が長い（半年や年単位）
  - 開発者という人間の個々とチーム化，日々の活動に関する心理的・社会的側面も含めた考慮が少ない
- 2001年のマニフェスト宣言（次頁）と
  そこから起因したアジャイルソフトウェア開発（Agile Software Development）の様々な動き

アジャイルマニフェスト

- プロセスやツールよりも個人と対話を
- 包括的なドキュメントよりも動くソフトウェアを
- 契約交渉よりも顧客との協調を
- 計画に従うことよりも変化への対応を

※ 左記のことがらに価値があることを認めながらも
私たちは右記のことがらにより価値をおく

[http://agilemanifesto.org/iso/ja/manifesto.html]

「背後にある原則」もぜひ
[http://agilemanifesto.org/iso/ja/principles.html]

アジャイルソフトウェア開発

- アジャイルソフトウェア開発（Agile Software Development）
  - 前述のマニフェストを踏まえ，顧客と連携して反復的・漸進的開発を行うアプローチの総称
- 反復的・漸進的開発（Iterative and Incremental Development）：2～3週間や2～3か月のサイクルでのリリースを反復する
  - 「価値が高い一部分から作り，そのフィードバックも踏まえ次に作ることも決めていく」ことの繰り返し
  - 対比：「要求を全て洗い出し，全て作ってから顧客へ」

典型的な原則の例（１）

- 都度開発対象となる機能や進め方を決めていく
  - 顧客も含めた定期的なミーティングを行い，達成すべきゴールやTODOのリスト，その優先度の管理を行っていく
- いつでもコードは全体が統合されており動作する
  - 部品だけの状態，統合して動かない状態にしない
  - 顧客にとって価値があるテストの観点から，システム全体が常に検査されており，顧客に価値を提供する状態を更新していく
  - そのために，構成管理と連動した自動テストや仮コード（モック）の活用などを行う

典型的な原則の例（２）

- チームは自己組織化されている
  - 一人の管理者が指示を出すのではなく，全体がプロジェクトの状況を把握し，各自が自分の仕事に対して最終決定権を持つ
- 最小限のソフトウェアの開発を行う
  - YAGNI : You Ain’t Gonna Need It
  - いつかわずかなユーザが使うような機能に振り回されないようにする
  - ドキュメント・モデルを作ることが手段ではなく目的

頻出する言葉（の一部）（１）

- プロダクトオーナー
  - 顧客の立場からのプロジェクト参加者
- ユーザーストーリー
  - 「講師である石川が学生の出欠状況を把握できるようにしたい．なぜならば・・・」といった簡潔で具体的な形式（誰，何，なぜ）で簡潔に，ユーザー視点で要求を明文化する
- スクラムマスター，コーチなど
  - チーム内外のファシリテーション，外部妨害対処などを行う役割を指す
  - プロジェクトを管理する権限を持つわけではない

頻出する言葉（の一部）（２）

- イテレーション・スプリント
  - 反復の単位，通常1週間から高々1か月
- バックログ
  - やりたいこと・やるべきことの一覧，「実際に今やること・やっていること」はその中の一部
  - プロダクトのバックログ，イテレーションごとのバックログ（スプリントバックログ）
- ベロシティ
  - 開発の速度（見積あるいは実績），適応的に次の目標や行動を決めるため

目次

- アジャイルソフトウェア開発概要
- アジャイルプラクティス

プラクティス

- プラクティス（Practice）
  - アジャイルソフトウェア開発は，原則とそれを達成するための実践的ノウハウ・技術の集合として実践していく
  - 開発活動に関するパターンであるとも言える
    （GoFはコード設計に関するパターンであった）
  - デザインパターンと同じく，「どういう状況・課題に対して，どういうアプローチをとるのか」を踏まえて，自分たちで採用するものを決めていく
- プラクティスの集まりとして方法論を定義
  - スクラムが代表的
  - ただ定義は弱く，組織によりカスタマイズされる

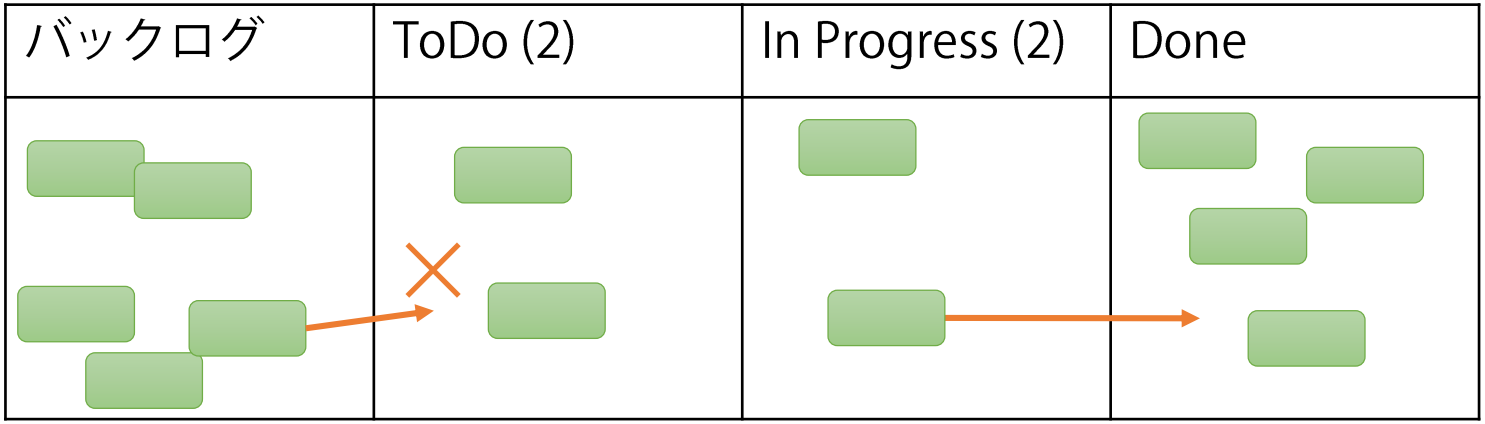
ユーザーストーリーマッピング

- ユーザーストーリーマッピング
  - 問題：プロダクトの全体像を共有するのが困難
  - 手段：利用者の時間を横軸に，優先順序を縦軸に，ユーザーストーリーを配置

図は [IPA, アジャイル型開発における
プラクティス活用 リファレンスガイド] より

プラクティスの例：かんばん

- かんばん
  - 問題：既存システムを運用しつつ拡張していく場合など，不確実・不定期に要求（の変化）が発生していく場合，扱う量を適切な範囲に制御し計画する必要がある
  - 手段：タスクの実施状況を管理するボードを用いるとともに，並行実施数の制限（WIP: Work in Progress）を明記・実施する

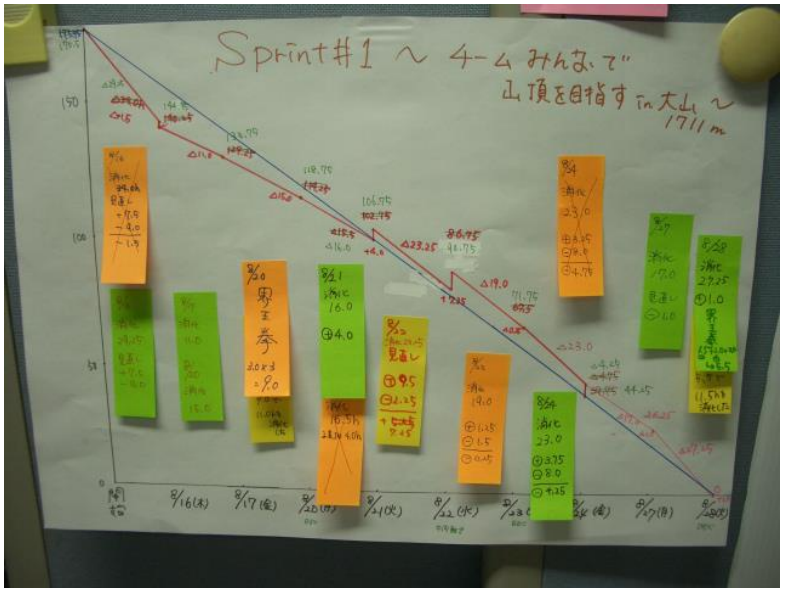


プラクティスの例：ベロシティ計測

- ベロシティ計測
  - 問題：イテレーションあたりの開発量が見積もれないと，リリース日を見積もれない
  - 手段：イテレーションあたりの完了ストーリー（ポイント）数を「ベロシティ」として計測し，見積に用いる
  - 留意点：新しいチームの場合複数イテレーションでベロシティを計測，必要に応じ従来の時間見積も用いる

プラクティスの例：バーンダウンチャート

- バーンダウンチャート
  - 問題：イテレーション内，あるいはリリースサイクル内において，実際の進捗状況を見て，適応的に行動を定めていく必要がある
  - 手段：横軸にリリースまでの時間軸，縦軸に作業量（時間やストーリーポイント）をとり，進捗状況（計画・実状）を可視化する



図は [IPA, アジャイル型開発における
プラクティス活用 リファレンスガイド] より

プラクティスの例：インセプションデッキ

- インセプションデッキ
  - 問題：顧客や様々なステークホルダーの間で，プロダクトの目的や方向性が不明確なことがある
  - 手段：10の質問により明確化
    - 例：われわれはなぜここにいるのか？
    - 例：やらないことリストを作る
    - 例：「ご近所さん」を探せ
    - 例：夜も眠れなくなるような問題は何だろう？
    - 例：何をあきらめるのかをはっきりさせる

プラクティスの例：プランニングポーカー

- プランニングポーカー
  - 問題：初期には開発対象の知識が少なく，参加者複数人の見解を考慮して，見積を行う必要がある
  - 手段：

1. 何か標準の見積を初期値とし，それに対する見解（「ずっと工数高いはず，+3」 など）を各自が一斉にカードで提示
2. 一番高い人・低い人が意見を述べ，全体で議論する
3. 見解が一致するまで繰り返す

  - 留意点：時間を決める，テスターなど様々な立場の人が加わるようにする，感覚的な小さい数字を用いる

プラクティスの例：ペアプログラミング

- ペアプログラミング
  - 問題：個人の知識・スキルには差異がある，個人で達成できる品質以上のものを作りたい，知識が個人から広がらない
  - 手段：プログラミング（に限らない）作業をペアで行う

プラクティスの例：テスト駆動開発

- テスト駆動開発（TDD : Test-Driven Development）
  - 問題：テストの定義・実施を後回しにすると，正しくないコードを書いていたり，既存のコード部分の正しさを壊してしまっていたりする
  - 手段：「実行可能なテストコードを定義し，それをパスするコードの開発を行う」ということを小さい単位で繰り返す
    （特に，「とにかくテストをパスするコードを書き，その後リファクタリングする」という思想も強い）

プラクティスの例：継続的インテグレーション

- 継続的インテグレーション（Continuous Integration）
  - 問題：開発者は各自の環境で小さな部品だけに取り組んでいることがあるが，環境設定などにより，組み合わせて全体として動かないことがよくある
  - 手段：定期的あるいはコミットの度に，必ずシステム全体をビルド，テストする（Jenkinsなどのツールにより自動化する）ことで，各自の作業がシステム全体とリンクし続けるようにする

プラクティスの例：その他

- 朝礼ミーティング
- ふりかえり
- コーディング規約
- 共通の部屋
- ニコニコカレンダー
- ビヘイビア駆動開発（Behavior-Driven Development : BDD）
- 受入れテスト駆動開発（Acceptance Test-Driven Development : ATDD）

アジャイルに対する問い

- 適用可能な状況
  - 少人数，物理的に一箇所
  - 各開発者が広い知識・スキルを持つ（多能工）
- 柔軟な変更に関する技術的な困難さ
  - 事前準備（変更を見越した設計）なしに，何でもすぐに要望に応じて変えられるわけではない
- ちょっと宣伝が宗教化（最近は少ない）
  - 従来（モデリング，ドキュメント，計画）の過剰な否定
- 従来のやり方との現実的な融合

今回の参考文献

- アジャイルイントロダクション (トップエスイー入門講座)
  - B. Meyer, 土肥ら訳，近代科学社，2018
- アジャイル型開発におけるプラクティス活用 リファレンスガイド
  - IPA, 2013
  - https://www.ipa.go.jp/sec/softwareengineering/reports/20130319.html

まとめ

- アジャイルソフトウェア開発
  - 計画・定型化への偏りへの反動，変化が激しい時代への対応として一つの主軸となる考え方へ