MSF Agile ver.4

Microsoft Solutions Framework
for Agile Software Development
ver. 4.x

問題

- **これから新しい開発プロジェクトが始まります**
  - マネージャに呼ばれた**あなた**は、こう言われました。「 だいたい 10人くらいの開発チームになるだろう。最初の 3人は、キミの自由に選んでいいよ。」
  - さて、あなたを含めて 4名、どんな基準で選びますか?
    - 要件定義からのスタートです。
    - あとから増えるメンバは、きっと大半が新兵と外人部隊です。

MSF Agile 的な回答

- なにを作ればいいか考えられる人
- どうやって作るかを考えられる人
- どうやったら壊せるかを考えられる人
- 上の 3人 + 顧客 + 自社 の調整をとれる人
  - 違うベクトルを向いた 3人 + 調整役 (PM)
  - 要件定義の段階から、作り方 (アーキテクト) も、壊し方 (テスター) も、同時に考慮する。

自己紹介

- **山本 康彦（ biac ）**
  - いまだにプログラムを書きたがる 50歳
  - http://nadia.kabe.to/
- **名古屋のとある ISV 勤務**
  - 現在、 WPF を使った業務アプリケーションの開発プロジェクトで品質保証を担当
  - MFS Agile を部分的に実施中
- **もとは機械設計者**
  - ものごとの見方・考え方がズレてるかも

Agenda

- MSF って何だろう?
  - スコープ、出自、種類
- MSF Agile って何だろう?
  - Agile なプロセス、CMMI と Agile
- Agile にやるにはチームワークが大事
  - チームモデル、提言者グループ、ロール
- Agile な実装は TDD で
  - ワークストリームとアクティビティ
  - Test Driven Devlopment の効果

Microsoft Solutions Framework

- **MSF はソリューションを作り出すプロセス**
  - アプリケーションを開発し、稼動させるまで。
  - 保守・運用には MOF（ Microsoft Operations Framework ）
  - Team Foudation Server のデフォルトは MSF
- **なぜ MS “プロセス” ではないのか?**
  - 「柔軟でスケーラブルなフレームワーク」を提供
  - 逆に言えば、 詳細はチームごと / プロジェクトごとに決めなければいけない。

MSF の簡単な歴史

- 1994 Ver.1
  - Microsoft 社内のベストプラクティスの集合体
- 2004 Ver.3.1
  - ホワイトペーパー等、MSF を公開（日本語文書も）
- 2006 夏 Ver. 4.0
  - VS2005 Team Foundation Server
- 2006 秋 Ver. 4.1
  - VSTS Database Professionals 追加に伴う改定
- 2007 末 Ver. 4.2
  - VS2008 Team Foundation Server

2つの MSF ver.4

- **MSF for Agile Software Development**
  - MSF CMMI の、(ほぼ) サブセット。
  - TDD が強く出ていたりするので、単純なサブセットというわけではなさそう。
  - チーム規模は 3人から、10人程度まで。
- **MSF for CMMI Process Improvement**
  - SEI CMMI (Capability Maturity Model Integration) Lv.3 の要件を満たす。
- **※ ver. 3 とは、かなり変わっている！**

Agenda

- MSF って何だろう?
  - スコープ、出自、種類
- **MSF Agile って何だろう?**
  - Agile なプロセス、CMMI と Agile
- Agile にやるにはチームワークが大事
  - チームモデル、提言者グループ、ロール
- Agile な実装は TDD で
  - ワークストリームとアクティビティ
  - Test Driven Devlopment の効果

アジャイルソフトウェア開発宣言

私たちは自らソフトウェア開発を行い、他人のソフトウェア開発を手助けすることで、ソフトウェア開発のより優れた方法を発見している。この仕事を通して、私たちは以下のことを重視するようになった。

- ***プロセスやツール*** *よりも* **個人と相互作用**
- ***包括的なドキュメント*** *よりも* **動作するソフトウェア**
- ***契約交渉*** *よりも* **ユーザとの協調**
- ***計画に従う*** *よりも* **変化に対応すること**

つまり、***左側の項目*** にも価値はあるが、**右側の項目**により多くの価値を見いだしている

※ 原文: http://www.agilemanifesto.org/
※ 2001年2月米国ユタ州にて開催されたアジャイルアライアンス会議にて採択された

MSF Agile も、この価値観に従っています

重量級とアジャイル

	重量級プロセス	アジャイル
プロセス定義	詳細にマニュアル化	細部はチームに委ねる
設計（情報伝達）	詳細に文書化	ユーザとの協調 チームメンバ間の協調 動作するコード
属人性	排除したい	依存してよい
マネージメント	（アジャイルよりは）容易	困難（見えにくい）

アジャイルな開発プロセスは、敏捷に (agile に) 変化に対応するため、チームとチームメンバの力量に大きく依存する。

そのため、属人性を排除する方向の重量級プロセスに比べると、マネージメントが難しいといわれる。
MSF Agile では、TFS を使った「見える化」で、マネージメントをサポートすることを推奨している。

MSF の プロセス ガイダンス

- MSF ver.4 の日本語ドキュメントは、プロセスガイダンスしかない (たぶん)。
- TFS に入っている。
  VS2008 TFS beta2 にも含まれている (たぶん、正規の評価版にも)
- 誤記・誤訳もあるので注意。

MSF Agile とは

- シナリオ主体で、状況に応じたアジャイルなソフトウェア開発プロセス（プロセスガイダンスの「原則」より）
- 9つの原則
  - *顧客とのパートナー関係
  - *共有ビジョンに向かっての作業
  - *段階的なデリバリ
  - *品質への投資
  - *チーム メンバの権限付与
  - *明確な説明責任の確立
  - *あらゆる経験から学ぶ
  - *開かれたコミュニケーションの促進
  - *機敏さを保ち変更に適応

MSF Agile の特徴

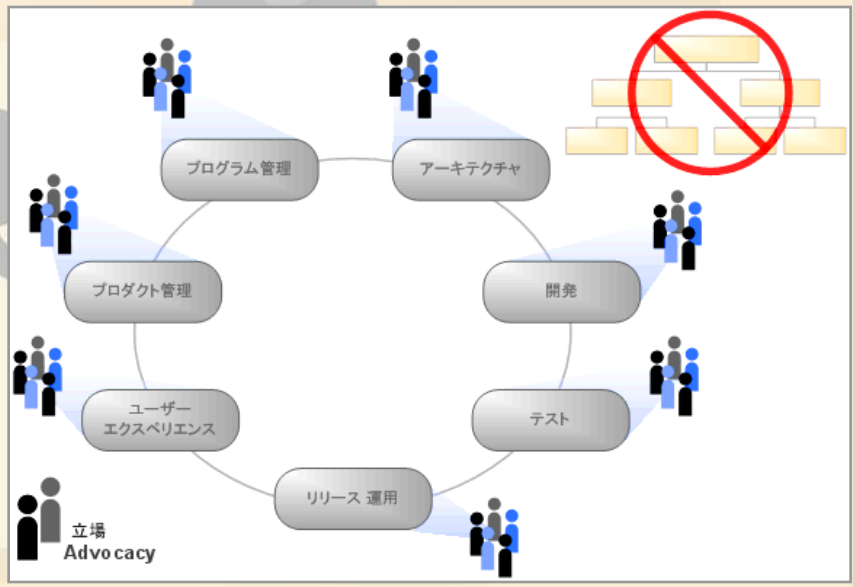
- **チームモデル**
  - チームモデルを明確に定義している
- **ペルソナ/シナリオ法**
  - 要件定義～外部設計は、ペルソナ / シナリオ法（ キャラ / 脚本法 ）を軸に据えている
- TDD
  - 実装 / デバッグには、Test Driven Development を取り入れている。

Agenda

- MSF って何だろう?
  - スコープ、出自、種類
- MSF Agile って何だろう?
  - Agile なプロセス、CMMI と Agile
- **Agile にやるにはチームワークが大事**
  - **チームモデル、提言者グループ、ロール**
- Agile な実装は TDD で
  - ワークストリームとアクティビティ
  - Test Driven Devlopment の効果

Team Model

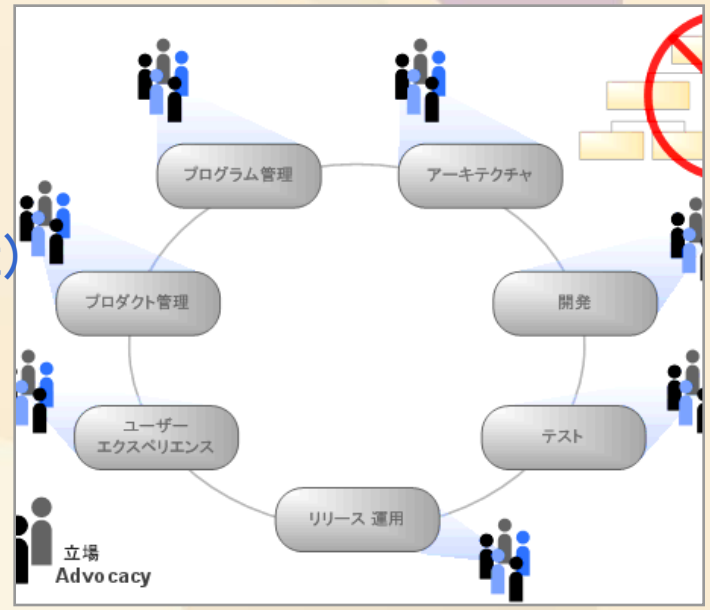
- 明確な意思疎通を図る ピア チーム (Team of Peers)
- プロジェクトの成功に貢献するすべての提言者 (Advocacy Group)
- プロジェクトの要件に応じた規模の調節



advocacy [ˈad-və-kə-sē]
主張。弁護。特に，権利擁護の主張。
advocacy journalism
特定の主義を擁護する報道機関
advocacy group
市民運動団体

7つの提言者グループ(Advocacy Groups)

- アーキテクチャ (Architecture)
  システム全体の仕掛けを代表する立場
- プロダクト管理 (Product Management)
  顧客ビジネスを重視する立場
- プログラム管理 (Program Management)
  ソリューションの納期を重視する立場
- 開発 (Development)
  技術解を重視する立場
- テスト (Test)
  顧客の視点からソリューションの品質を重視する立場
- ユーザー エクスペリエンス (User Experience)
  対象ユーザーにとって最も効果的なソリューションを重視する立場
- リリース運用 (Release Management)
  適切なインフラストラクチャへのソリューションの円滑な配置を重視する立場



提言者グループの兼任 (1)

	アーキテクチャ	プロダクト管理	プログラム管理	開発	テスト	ユーザーエクスペリエンス	リリース管理
アーキテクチャ	-	×	○	○	△	△	△
プロダクト管理	×	-	×	×	○	○	△
プログラム管理	○	×	-	×	△	△	○
開発	○	×	×	-	×	×	×
テスト	△	○	△	×	-	○	○
ユーザーエクスペリエンス	△	○	△	×	○	-	△
リリース管理	△	△	○	×	○	△	-

※ ○:可能 / △:普通はしない / ×:避けるべき

提言者グループの兼任 (2)

- 三権分立 +1

[三権を調整する立場]
・プログラム管理 (PM)

[顧客を代弁する立場]
・プロダクト管理
・ユーザーエクスペリエンス

(設計)

[モノを構築する立場]
・アーキテクチャ
・開発

(開発)

[品質を保証する立場]
・リリース管理
・テスト

(テスト)

ロール（役割分担）

- MSF Agile では、6つのロール

V4.1 から、DB 関係の
ロールが 2つ追加された。

[三権を調整する立場]
・プロジェクト マネージャ

[顧客を代弁する立場]
・ビジネス アナリスト
(設計)

[モノを構築する立場]
・アーキテクト
・開発者
(・DB 開発者)
(開発)

[品質を保証する立場]
・リリース マネージャ
・テスター
(・DB 管理者)
(テスト)

Agenda

- MSF って何だろう?
  - スコープ、出自、種類
- MSF Agile って何だろう?
  - Agile なプロセス、CMMI と Agile
- Agile にやるにはチームワークが大事
  - チームモデル、提言者グループ、ロール
- Agile な実装は TDD で
  - ワークストリームとアクティビティ
  - Test Driven Devlopment の効果

ワークストリーム と アクティビティ (1)

旧来の工程	ワークストリーム	アクティビティ	ロール
要件定義・外部設計	プロジェクト ビジョンの捕捉	1. ビジョン ステートメントの作成 2. ペルソナの定義 3. ペルソナの改善	ビジネス アナリスト
	シナリオの作成	1. シナリオのブレーンストーミング 2. ライフスタイル スナップショットの開発 3. シナリオ リストの優先度の決定 4. シナリオ記述の作成 5. シナリオのストーリーボードの作成	ビジネス アナリスト
	サービス品質要求の作成 必要に応じて ・画面定義 ・帳票定義	1. サービス品質要求のブレーンストーム 2. ライフスタイル スナップショットの開発 3. サービス品質要求リストの優先度の決定 4. サービス品質要求の作成 5. セキュリティの目標の特定	ビジネス アナリスト

※ ペルソナ/シナリオ法 : コンピュータは、むずかしすぎて使えない! (アラン クーパー 978-4881358269)

ワークストリーム と アクティビティ (2)

旧来の工程	ワークストリーム	アクティビティ	ロール
内部設計・実装・単体テスト	ソリューション アーキテクチャの作成	1. システムのパーティション化 2. インターフェイスの決定 3. 脅威モデルの開発 4. パフォーマンス モデルの開発 5. アーキテクチャ プロトタイプの作成 6. インフラストラクチャ アーキテクチャの作成	アーキテクト
	開発タスクの実施	1. 開発タスクのコスト計算 2. 単体テストの作成またはアップデート 3. 開発タスクのコード作成 4. コードの分析 5. 単体テストの実行 6. コードのリファクタリング 7. コードのレビュー 8. コード変更の統合	開発者
	製品のビルド	1. ビルドの開始 2. ビルドの検証 3. ビルドの修復 4. ビルドの承認	開発者

ワークストリーム と アクティビティ (3)

旧来の工程	ワークストリーム	アクティビティ	ロール
結合テスト ～ テストの実施	シナリオのテスト	1. テスト方法の定義 2. 妥当性確認テストの作成 3. テスト ケースの選定および実行	テスト担当者
	サービス品質要求のテスト	1. テスト方法の定義 2. パフォーマンス テストの作成 3. セキュリティ テストの作成 4. ストレス テストの作成 5. 負荷テストの作成 6. テスト ケースの選定および実行	テスト担当者
結合テスト ～ 欠陥追跡	シナリオのテスト	4. バグの登録 5. 予備テスト (探索的テスト) の実施	テスト担当者
	サービス品質要求のテスト	7. バグの登録 8. 予備テスト (探索的テスト) の実施	テスト担当者
	バグの終了	1. 修復を検証する 2. バグの終了	テスト担当者

exploratory testing

ワークストリーム と アクティビティ (4)

旧来の工程	ワークストリーム	アクティビティ	ロール
結合テスト ～ 欠陥修正	**バグの修正**	1. バグの再現 2. 単体テストの作成またはアップデート 3. バグの原因の特定 4. バグの再割り当て 5. バグ修正ストラテジに基づく判断 6. バグ修正のコーディング 7. 単体テストの実行 8. コードのリファクタリング 9. コードのレビュー 10. コード変更の統合	開発者

ワークストリーム と アクティビティ (5)

	ワークストリーム	アクティビティ	ロール
旧来の工程 プロジェクト管理	イテレーションの計画	1. イテレーションの長さの定義 2. シナリオの見積り 3. サービス品質要求の見積り 4. シナリオのスケジュール 5. サービス品質要求のスケジュール 6. バグ修正作業の割り当て 7. シナリオのタスク配分 8. サービス品質要求のタスク配分	プロジェクトマネージャ
	プロジェクトの管理	1. 目標の確認 2. 進捗状況の評価 3. テスト測度のしきい値の評価 4. バグのトリアージ 5. リスクの特定	プロジェクトマネージャ
	イテレーションの管理	1. イテレーションの監視 2. リスクの軽減 3. 振り返りの実施	プロジェクトマネージャ
	製品のリリース	1. リリース計画の実行 2. リリースの妥当性確認 3. リリース ノートの作成 4. 製品の配置	リリース マネージャ

イテレーション

- **イテレーション = 2週間 ～ 1ヶ月**
  - 開発期間をイテレーションで区切る。
  - イテレーション内で、 複数のワークストリーム。
  - 詳細な計画は、イテレーションごとに実施。
- **初期のイテレーション**
  - 要件定義～外部設計に重点（ 実装～テストは無いかも ）
- **後期のイテレーション**
  - 実装～テストに重点（ 要件定義～外部設計は無いかも ）

TDD の品質向上効果 (1) : バグ削減

- **設計書レビュー効果 = 30%**
  仕様が不明瞭なままでは、単体テストが書けない。単体テストを書いていると、仕様の不備に気付く。
- **単体テスト実施効果 = 30%**
  実装したコードは、必ず単体テストを実施されている。
- **トータルで約50% のバグ削減が見込める。**

※「レビュー / テスト 1段の実施で、欠陥の 30% が見つかる」
　1990年代の米国のデータ。
「ソフトウェア見積りのすべて」Capers Jones 著 / ISBN:4320097319 より。
　※※ 日本では、もう少し良いのでは?

TDD の品質向上効果 (2) : 設計

- **先にテストを考える = インターフェースの設計**
  一般に、クラスやメソッドのインターフェースをきちんと設計するのがよいとされている。
- **自動化されたテストがある = リファクタリング可能**
  動いているコードは触るな、と言われてきた。自動でテストできるなら、リファクタリングしても動作は変わっていないことを簡単に保証できる。

ありがとうございました

- MSF って何だろう?
  - スコープ、出自、種類
- MSF Agile って何だろう?
  - Agile なプロセス、CMMI と Agile
- Agile にやるにはチームワークが大事
  - チームモデル、提言者グループ、ロール
- Agile な実装は TDD で
  - ワークストリームとアクティビティ
  - Test Driven Devlopment の効果
- http://akari.kabe.co.jp/magsite/List.modf?s=bwMSF