模範的 Scrum Master チェックリスト

あなたはフルタイムのファシリテーターかな？

"まずまずな"Scrum Master は、一度に 2 つか 3 つのチームを扱うことができる。もしあなたが、スクラムマスターの役割を「会議の進行」、「タイムボックスの施行」、「不必要なレポーティングなどの『障害物』への対応」に限定することに満足するなら、パートタイムとしてうまくやってくことはできるだろう。 チームはそれでも、平均点や Scrum を導入する前の期待値は超えるだろうし、そして多分何も破滅的な事は起きたりしない。

でも、もし、これまでに無いような事を成し遂げて素晴らしい時を過ごせるチームを心に思い描けるなら-- 組織の改革も含めて --、"卓越した"Scrum Master になることを考えてみよう。

"卓越した"Scrum Master は、一度に一つのチームだけを扱うことができる。

私達は、特に Scrum を開始する段階においては、一つのチームに一人の献身的な Scrum Master を置くことをお勧めする。

もしあなたが何もやることを見つけられないなら、あなたの Product Owner、チーム、チームのエンジニアリング、そしてチームの外の人びとの声に耳を傾けてみよう。 特効薬なんかどこにも無いんだけど、私が今まで見てきた Scrum Master たちがチームの何を見守っているのか、概要をまとめてみた。

Product Owner はどうしているかな？

あなたは、プロダクトバックログやリリースプランの維持管理を手伝うことで、Product Owner の成果を進歩させる。 (Product Owner だけがバックログの優先度を決定できることに注意)

- プロダクトバックログは、彼・彼女の最新の考えに基づいて優先度付されているかな？
- バックログには、すべてのステークホルダの要望・要求が反映されているかな？バックログは"表に出ているもの"だってことを覚えておこう。
- プロダクトバックログは扱いやすいサイズかな？扱いやすい数にしておくために、細かく具体的になったものを一番上に、ふわっとしたものを一番下にしておこう。一番上から遠く離れたバックログをやたらと分析するのは逆効果だ。開発している間に要求は変わっていくものだからね。

- バックログにある要求は(特に一番上に近いもの)「INVEST(independent, negotiable, valuable, estimable, small, and testable - See more at: https://www.scrumalliance.org/community/articles/2010/november/an-example-scrummaster-s-checklist#sthash.YiOwcsXJ.dpuf)」 な形式になっているかな？
- Product Owner に、技術的負債についてと、それを避ける方法について教えたことはあるかな？自動テストとリファクタリングは、全てのバックログアイテムの Done の定義のピースとして追加しておけるかもしれない。
- バックログは情報ラジエーターとして、全てのステークホルダーから丸見えになっているかな？
- バックログの管理になにか自動ツールを使っているとしたら、本当にそれはみんなの役に立っているかな？自動的管理ツールは、Scrum Master による熱放射が無いと、情報冷蔵庫になってしまう危険性を秘めているんだ。
- プリントアウトしてみんなに見える状態にするのを手伝えるかな？
- 大きな図を描いてみんなに見える状態にするのを手伝えるかな？
- Product Owner がバックログアイテムを適切なリリースに分割するのを手伝っているかな？(1 発め、2 発め、倉庫行き、とか)
- 全てのステークホルダ(チームも含む)が、リリースプランが現実味を帯びていて実際のベロシティ(例：スプリントごとの消化ストーリーポイントとか)にそっている事を知っているかな？
- Product Owner は、前回のスプリントレビューミーティングの後でリリースプランを調整したかな？適切にテストされた納品可能なプロダクトを毎スプリント間に合わせ、スプリントごとにリリースを再計画し、より重要な仕事のためにいくつかを先延ばしできる Product Owner は多くない。Mike Cohn のプロダクト/リリースバーンダウンチャートを、スプリントレビューごとに見せるといいかもしれない。スコープやスケジュールの漂流が見つかりやすくなるだろう。

チームはどうしているかな？

- チームは、ほとんどの時間を Flow 状態ですごせているかな？この状態の特徴はこうだ：
    - 明確なゴール(期待値とルールが分かり、ゴールが到達可能で、かつ各自のスキルセットと力量に合致している);
    - 集中し、意識すること、限られた注目の中での高いレベルの集中；
    - 自意識が無い状態、行動と気付きが合致すること；

    - 歪んだ時間への意識(時間が早くすぎるかのような感覚)；
    - 直接・即時フィードバック(活動の中で成功と失敗が明白で、必要な振る舞いがすぐできる)；
    - 能力とチャレンジが良いバランス(簡単すぎもせず、難しすぎもしない)；
    - 状況や活動を踏まえて自分でコントロール出来るという意識；
    - 活動それ自体にやる価値があり、重い腰を上げる必要が無い；
- チームメンバーはお互いが好きで、いっしょにサボったり、お互いの成功を祝ったりしているかな？
- チームメンバーはお互いに高い道徳を維持するようにし、お互いに成長に向けて挑戦しているかな？
- 気まずくて議論していないような問題はあるかな？この本を読もう：http://www.amazon.co.jp/Crucial-Conversations-Tools-Talking-Stakes/dp/0071401946 もしくは、気まずい会話を気分よく変える事の出来るプロフェッショナルなファシリテーターを募集することを考えよう。
- いろんなフォーマットのスプリントレトロスペクティブを試したかな？この本を読もう：http://www.amazon.com/Agile-Retrospectives-Making-Teams-Great/dp/0977616649
- チームはアクセプタンスクライテリアに焦点を合わせているかな？もしかしたらスプリントの途中でアクセプタンスクライテリアをチェックしなおす機会を持ったほうがいいのかもしれない。
- スプリントバックログはチームが今何をしているかを反映しているかな？スプリントのゴールに向かうのを邪魔したり気を散らしたりするものは"Impediment"だよ。
- チームは 3–9 人で構成され、出荷可能なプロダクトを作るために必要な全ての能力を持ち合わせているかな？
- チームのタスクボードや見積もりは最新状態になっているかな？
- チームの自己管理のためのもの(タスクボード、スプリントバーンダウンチャート等)はチームに見える状態で、チームにとって便利かな？
- チームの自己管理のためのものは、マイクロマネージャから適切に守られているかな？
- チームメンバーは自ら進んでタスクに取り組んでいるかな？
- 技術的負債返済リスト(チームのベロシティを奪うもの)がバックログに入っているか、Product Owner とそれについて話をしているかな？
- チームメンバーは肩書をチームの外に捨てているかな？

- チーム全体として、自ら正しくテストやユーザードキュメントの事を考慮しているかな？

エンジニアリングはどうかな？

- 開発中のプロダクトは「テスト実行」ボタンがあって、だれでも(チームも別チームも)壊したら簡単に検出出来るようになっているかな？ だいたいは xUnit で実現されているはずだ。
- 自動化された End-to-end のテスト(機能テスト、と呼ばれていたもの)とユニットテストは適切なバランスになっているかな？
- チームは、システムを開発するのと同じ言語で、機能的テスト・単体テストを書いているかな？(専用のスクリプト言語やキャプチャツールとかを使うのではなく)　可能だよ。
- チームは、システムテストとユニットテストの間に便利なグレーゾーンを見つけているかな？
- CI サーバは、リグレッションテストでクラッシュを発見したときに速やかにアラートが鳴るかな？("日毎のビルドは弱虫のためのものだ" ケント・ベック)
- 全てのテストは CI サーバの結果を巻き込んでいるかな？
- チームは、最初に行う大げさな設計に代わって、連続的なデザインと容赦無いリファクタリングに喜びを見出しているかな？リファクタリングには厳密な定義がある：外から見た振る舞いを変えずに、内側の構造を変えることだ。リファクタリングは 1 時間の間に何度でも、コードの重複、複雑な条件のロジック(長くてインデントが深すぎるコード等)、おかしな名前の識別子、結合の深すぎるオブジェクト、あまりにも多すぎる責務を持ったオブジェクト、等を見つけたらいつでもやるものだ。自動化テストカバレッジによって自信を持ってリファクタリングを実践出来る。テストカバレッジの穴や放置されたリファクタリングは、技術的負債と呼ばれる癌だ。
- 各プロダクトバックログアイテムの Done の定義は、完全な自動テストカバレッジとリファクタリングを含んでいるかな？毎スプリントで納品可能なプロダクトを作ろうと思ったら、eXtreme Programming(ケント・ベック、ロン・ジェフリーズの)を学ばないわけにはいかないよ。
- チームメンバーは主にペアプログラミングに時間を費やしているかな？ペアプログラミングは劇的にコードのメンテナンス性を向上し、バグ率を減らしてくれる。でも人間の限界に挑戦することになるし、より時間を費やしているように見える(もし品質より量を重視するなら)。みんなに強制するよりは、

まずは自分で何かペアで作業をしてみることでメンバーを導こう。何人かはこのやり方を気に入って始めてくれるだろう。

組織はどうかな？

- チーム内協調の量は適切かな？スクラムオブスクラムはこれを行うのに唯一の方法だ。また、他のチームのスタンドアップミーティングに大使を送り込むことを考えてもいいだろう。
- チーム内協調は、推奨されている通り、実際に手を汚している人同士で行われているかな？
- Scrum Master 同士で会って、組織の妨害リストに取り組んでいるかな？
- 可能であれば、組織の妨害リストは開発組織の重役の部屋の壁に貼ってあるかな？お金に換算されたコスト、マーケット投入の遅れの時間、失った品質、失った顧客機会、などはどうかな？(でも Ken Schwaber の発見："死んだ Scrum Master に居場所はない"を覚えておこう)
- あなたの組織は、所属する複数のチームの集合的なゴールに向かう為のキャリアパスの一つになっているかな？もしテストや、テスト自動化、ユーザードキュメントを書くこと等を犠牲にしてでもプログラミングやアーキテクチャとしての仕事をやることに動機があるようであれば、答えは No だ。
- あなたは"学ぶ組織"を作ることを助けているかな？
- あなたの組織は、経済誌や独立した情報誌等で最も働くのに適した場所だと認知されているかな？

恐怖はあなたの友達

一度あなたが変化を起こせるということを悟ったなら、それを行う事を怖がるだろう。それはあなたが正しい道にいることのサインだ。 -- MJ