スクラムマスターチェックリスト

Michael James (mj@seattlescrum.com)
14 September 2007 (Revised 2 Feb 2016)
http://ScrumMasterChecklist.org
翻訳： 濱千代 直也

フルタイムで関わっていますか？

“まずまずな” スクラムマスター は、一度に2つまたは3つのチームを受け持ちます。 スクラムマスター の役割を「ミーティングの主催」、「タイムボックスの管理」、および「チームから報告された問題への対処」に限定すれば、パートタイマーながらなんとか上手くやることができるでしょう。 そのような状態であってもきっと、チームは組織のベースライン（スクラム前の期待値）を上回っているでしょうし、この先も致命的な問題は発生しないでしょう。

しかし、前人未到な出来事を成し遂げる素晴らしいチームをイメージすることができるなら、（変革した組織の中で）”卓越した” スクラムマスター になれる可能性があります。

“卓越した” スクラムマスター は一度に1つのチームだけを受け持ちます。

スクラム の導入時期においては特に、献身的なスクラムマスターをチーム毎に1人用意することをお勧めします。

特にやることが無いなら、「プロダクトオーナー」「チーム」「開発プロセス」「（チーム外の）組織」を観察してみましょう。 誰にでも効く特効薬では無いですが、スクラムマスターが見落としがちなポイントをまとめてみました。以下のそれぞれのボックスに、√, Δ, ?, もしくは N/A をつけてみてください。チェックの付け方は一番最後のページに記載しています。

Part I -- プロダクトオーナー はどうしてますか?

スクラムマスターがプロダクトバックログやリリースプランの調整の仕方に対してアドバイスなどを行えば、プロダクトオーナーはより効果的に動けるようになるでしょう。(注意：バックログの優先順位づけに対して責任を持つのは、プロダクトオーナーです。)

☐ プロダクトバックログは、プロダクトオーナーの最新の考えに基づいて、優先順位付けされていますか？

☐ 全てのステークホルダーの要求事項がプロダクトバックログに取り込まれていますか？ （注意：バックログは “徐々に明らかになるもの” です。）

☐ プロダクトバックログは管理可能なサイズですか？ PBI（プロダクトバックログアイテム）の数を管理可能な数に調整し、具体的なものを上に、曖昧なものは下にするようにしよう。 あまりにも下位にあるPBIを具体化するのは無駄です。 なぜなら、ステークホルダーや顧客がプロダクトに触れた結果生まれたフィードバックを受けて、あなたの要求は変わるからです。

- [ ] 要件（特に、バックログの上にある要件）は、INVESTなユーザーストーリー[1]として表現できていますか？ INVESTとは、Independent（独立している）・Negotiable（交渉可能である）・Valuable（価値がある）・Estimable（見積可能である）・Small（小さい）・Testable（テスト可能である）であることです。

- [ ] プロダクトオーナーに「技術的負債とは何か」「技術的負債を避ける方法」についてレクチャーしていますか？ 1つの解決策としては、自動テストとリファクタリングを、PBIの「Done」の定義に取り込むことです。

- [ ] バックログは情報の発信源となっていて、すべてのステークホルダーがいつでも見られる状態になっていますか？

- [ ] バックログ管理に電子ツールやソフトウェアを使用している場合、誰でも簡単に利用できる状態になっていますか？ 電子ツールやソフトウェアは、スクラムマスターからの積極的な発信が必要な”情報冷蔵庫”になる危険性があるため、注意してください。

- [ ] 全員にプリントを見せて、情報発信を支援することができますか？

- [ ] 大きくて目に見えるチャートを作成して、情報発信を支援することができますか？

- [ ] プロダクトオーナーがPBIを適切に仕分できるように、支援をしましたか？仕分けには、「どのタイミングでリリースすべきか」「優先順位づけをすべきか」があります。

- [ ] このままでリリース計画が実現するかどうかについて、全員が知っていますか？スプリントレビューでプロダクトが「Done」する度に、プロダクト/リリースバーンダウンチャート[2]を全員で見るようにしましょう。 実際に完了したPBIのレートと、追加された新しいPBIの両方を示すチャートは、スコープ/スケジュール調整の必要性を早期に発見できるようにします。

- [ ] プロダクトオーナーが最後のスプリントレビューの後にリリース計画を調整しましたか？ 十分にテストされたプロダクトを予定通り出荷する数少ないプロダクトオーナーは、スプリントごとにリリースを再計画します。 より重要な作業が見つかれば、将来のリリースのためにいくつかの作業を後回しに必要があります。

Part II -- 開発チームはどうしてますか?

チームメンバーの協働を推進することがスクラムマスターには求められますが、技術的な作業においてはスクラムマスターが上手く推進できない可能性もあります。 スクラムマスターが担うべき、チームに対しての主要な責任について考えてみましょう。

- [ ] チームは”フロー状態”にありますか？ “フロー状態”[3]には以下のような特徴があります。:
  - 明確な目標（「望ましいこと」と「必須なこと」が識別可能である。達成可能な目標である。目標が自分のスキルセットと能力に対して適切である。）

---

[1] http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

[2] Mike Cohn, 「アジャイルな見積りと計画づくり ~価値あるソフトウェアを育てる概念と技法~」. (2005).

[3] Mihaly Csikszentmihalyi, 「フロー体験 喜びの現象学」 (1990).

- 集中と選択、フィールドを特定した上での集中力の高さ。
- 自意識が薄くなり、行動と意識が融合する。
- 直接的かつ即時のフィードバック（活動の過程で成功と失敗が明らかになるため、必要に応じて行動を調整することができます）。
- 能力レベルと挑戦のバランス（簡単でも難しくもない難易度）。
- 状況や活動を超えて自分をコントロールしてる感覚。
- 本質的に価値があるため、すぐに行動に移れます。

☐ チームメンバーはお互いが好きで、一緒にバカができて、お互いの成功を祝っていますか？

☐ チームメンバーはお互いに、高いレベルで責任を持ち、成長を目指して挑戦していますか？

☐ 議論するのを極端に不快に思うあまり、議論を避けているような問題はありますか？[4]

☐ スプリントレトロスペクティブにおいて、いろんな場所ややり方を試しましたか？[5]

☐ チームはスプリントゴールをずっと大切にしてきましたか？ もしかしたら、このスプリントでコミットされたPBIの受け入れ基準の見直しとして、スプリント内点検を実施しても良いかもしれません。

☐ スプリントのタスクボードはチームが実際に行っていることを反映していますか？ 一日では終わらないタスクや隠されたタスクの「ダークマター」に注意してください。 スプリントゴールに関連しないタスクは、スプリントゴールに対する障害です。

☐ あなたのチームは、出荷可能なプロダクトインクリメントを構築するのに十分なスキルを備えた3〜9人のメンバーで構成されていますか？

☐ あなたのチームのタスクボードは最新の状態になっていますか？

☐ チームが自己管理してる成果物は、チームから見えるようになっていますか？チームにとって使いやすい状態ですか？

☐ これらの成果物は、チーム外のお節介焼き達から保護されていますか？ チーム外の人による過度なチェックは、チーム内部の透明性と自己管理を妨げる可能性があります。

☐ チームメンバーは自主的にタスクを行なっていますか？

☐ 技術的負債を返済する必要性が”Done”の定義に明示的に示されており、徐々にコードをより楽しい場所にできていますか？

☐ チームメンバーは、合意された作業（テスト、ユーザードキュメントなど）のすべての側面を包括的に担当するように、チームルームのドアにチームの役割を張り出してますか？

---

[4] Marshall Rosenberg, 「NVC 人と人との関係にいのちを吹き込む法 新版」(2003). もしくは、不快な議論をより快適にすることができるプロのファシリテーターの起用を検討してください。

[5] Derby/Larson 「アジャイルレトロスペクティブズ　強いチームを育てる「ふりかえり」の手引き」 (2006).

Part III -- 開発プロセスはどうなってますか?

- [ ] 開発中のシステムには誰でも（同じチームはもちろん、別のチームでも）、回帰テストの失敗（前は動いていた機能が壊れた）が簡単に検出できる「プッシュテスト」ボタンがありますか？ 通常、これはxUnitフレームワーク（JUnit、NUnitなど）で実現されます。

- [ ] 自動エンドツーエンドシステムテスト（機能テストなど）と自動単体テストとで、適切なバランスが取れていますか？

- [ ] チームは開発システムと同じ言語でシステムテストとユニットテストの両方を書いていますか？ そのツールだけで使うスクリプト言語や、チームの一部のみがメンテナンス方法を知っているようなツールでは、コラボレーションが強化されることはありません。

- [ ] チームは、システムテストと単体テストの間にある、重要でグレーな領域を明らかにしましたか？[6]

- [ ] 回帰テストが失敗した時、CI（継続的インテグレーション）[7]サーバーは自動的に警告を通知しますか？ また、このフィードバックループを数時間または数分間に短縮できますか？ （「Daily builds are for wimps.（毎日のビルドは弱虫のためのものです）」 - Kent Beck）

- [ ] すべてのテストをCIサーバーの結果に集約してますか？

- [ ] BDUF（Big Design Up Front：ソフトウェア全体を詳細に設計し終えてから、ソースコードの1行目を書き始めること）の代替案として、継続的な設計とリファクタリング[8]の喜びがあることに気づきましたか？ リファクタリングには厳密な定義があります。外部から見た挙動を変更せずに内部構造を変更することです。 重複したコード、複雑な条件付きロジック（過剰なインデントや長いメソッドはその兆候です）、不適切な名前の識別子、オブジェクト間の過度の結合などがある場合は、リファクタリングを1時間に何度も実施する必要があります。 リファクタリングを怠ると、将来的に製品を変更することが難しくなります。特に、悪いコードを扱う良い開発者を見つけるのが難しいです。

- [ ] PBIの "Done"の定義には、完全自動化されたテストカバレッジとリファクタリングが含まれていますか？ TDD（テスト駆動開発）を学習することで、これを実現しやすくなるでしょう。

- [ ] ペアプログラミングを実施できていますか？ ペアプログラミングにより、コードの保守性が大幅に向上し、バグ率が低下する可能性があります。 ペアプログラミングは人々の境界に挑戦し、時には時間がかかるように見えます（出荷可能な機能ではなく、コードの行数で測定した場合）。 例えば、最初からペアワークをチームでしていたとします。 そうすると、何人かはペアワークを好むようになるでしょう。

---

[6] http://blogs.collab.net/agile/junit-is-not-just-for-unit-testing-anymore

[7] http://www.martinfowler.com/articles/continuousIntegration.html

[8] Martin Fowler, 「リファクタリングー既存のコードを安全に改善するー」(1999).

Part IV -- チーム外の組織はどうしてますか?

- [ ] チーム間コミュニケーションの量は適切ですか？ 「Scrum of Scrum」はこれを達成するための方法の一つではありますが、多くの場合において最善の方法ではありません。[9]

- [ ] チームは独立して動くソフトウェアを作成できますか？ 技術領域をまたいだチームになっていますか？[10]

- [ ] 複数のスクラムマスターがいる場合、お互いに会話し、組織の課題を一覧化していますか？

- [ ] 組織上の問題はマネージャーのオフィスの壁に貼り付けられていますか？ 市場投入までに浪費した時間や、失った品質や、顧客機会の喪失などのコストを、お金で数値化することはできますか？ （Ken Schwaberの失敗から学びましょう：「A dead ScrumMaster is a useless ScrumMaster.（死んだScrumMasterは役に立たないScrumMasterです）」[11]）

- [ ] チームの目標と、組織の人事制度における評価基準は一致していますか？ テスト、テスト自動化、ユーザードキュメントを犠牲にして、プログラミングやアーキテクチャーの仕事をすることが評価[12]されるような組織であるなら、答えは「いいえ」でしょう。

- [ ] あなたの組織は、業界で最も優れた職場の1つ、または業界のリーダーとして、様々なメディアを通じて認知されていますか？

- [ ] “学習する組織”を作っていますか？

結論

上記チェック項目のほとんどをチェックしても、まだ時間が残っている場合は、ぜひあなたのお話を聞かせて欲しいです。

人間の創意工夫を生み出す公式はどこにもありません。 このチェックリストに列挙したポイントは、あなたの役立つかもしれないし、役に立たないかもしれません。

差をつけるためにあなたができることに気づいたのに、それを実行に移すのに恐怖を感じているかもしれません。 ただ、その恐怖は、あなたが正しい道を歩み始めた証拠です。

---

[9] http://less.works/less/framework/coordination-and-integration.html を見て、他の方法を確認しましょう。

[10] http://FeatureTeamPrimer.org/

[11] Ken Schwaber, 「スクラム入門-アジャイルプロジェクトマネジメント」 (2004)

[12] Alfie Kohn, 「報酬主義をこえて」 (1999)

組織改善フォーム

表出した課題:

根本原因 （「なぜ」を5回繰り返そう）:

ビジネスへの影響:

感情面への影響:

明確で行動可能なアクション:

組織改善フォーム

表出した課題:

根本原因 （「なぜ」を5回繰り返そう）:

ビジネスへの影響:

感情面への影響:

明確で行動可能なアクション:

組織改善フォーム

表出した課題:

根本原因 （「なぜ」を5回繰り返そう）:

ビジネスへの影響:

感情面への影響:

明確で行動可能なアクション:

組織改善フォーム

表出した課題:

根本原因 （「なぜ」を5回繰り返そう）:

ビジネスへの影響:

感情面への影響:

明確で行動可能なアクション:

組織改善フォーム

表出した課題:

根本原因 （「なぜ」を5回繰り返そう）:

ビジネスへの影響:

感情面への影響:

明確で行動可能なアクション:

組織改善フォーム

表出した課題:

根本原因 （「なぜ」を5回繰り返そう）:

ビジネスへの影響:

感情面への影響:

明確で行動可能なアクション:

説明書

このチェックリストをトレーニングとして渡された場合、もしくは現在の（または最新の）あなたの雇用者がスクラムのようなものを試している場合は、「あなたから見えた状態」でチェックをつけてください。 各項目に次のいずれかを記入してください：

√　(既に実現できている)

Δ　(改善できそうだし、方法も分かる)

?　(改善できそうだけど、方法が分からない)

N/A　(適用できない、メリットがない)

現在の（または最新の）あなたの雇用主がスクラムのようなものを試していない場合は、各項目に次のいずれかを記入してください：

√　(既に実現できている、簡単に実現できそう)

Δ　(挑戦できそうだし、方法も分かる)

?　(挑戦できそうだけど、方法が分からない)

N/A　(適用できない、メリットがない)

全てにチェックをつけたら、このチェックリストに関係あるかどうかに関わらず、組織改善フォームに2〜6個の組織課題を宣言しましょう。 その中で1%でも改善する可能性がある組織課題を選んでください。