スクラムマスターチェックリスト

Michael James (mj@seattlescrum.com)

14 September 2007 (Revised 8 Feb 2019)
http://ScrumMasterChecklist.org

翻訳： 知花 里香

あなたはチームのフルタイムのファシリテーター（促進者）ですか？

典型的なスクラムマスターは、１度に２～３チームを担当します。
もしあなたがスクラムマスターの役割を、ミーティングの運営やタイムボックスの意識の促進、周囲の人たちから共有される明らかな障害物を取り除いたりする、ということで十分だと判断するのであれば、パートタイム感覚でもこなすことができます。
チームはそれでも組織の期待を超えられるでしょうし、目も当てられないほど悲惨な状況にはならないでしょう。
一方で、誰もが不可能と思うようなことをチームが成し遂げ、組織変革までを見据えられるなら、あなたは”卓越した”スクラムマスターであると考えられます。

”卓越した”スクラムマスターは、一度に１つのチームを担当します。１人の専任スクラムマスターにつき、チーム7名程度から始めることをお薦めします。
もしどのようなことをしたらいいのかイメージがもてない場合は、プロダクトオーナーやチーム、チーム外の組織がどのような考えをもっているか、今どのようなことが起きているのか、チームの技術や開発方法はどうなっているか、理解することから始めましょう。

万能な処方箋ではありませんが、私がこれまで出会ってきたスクラムマスター達が見落としがちなことをまとめました。
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。

パート1: プロダクトオーナーはどのように過ごしていますか？

スクラムマスターは、プロダクトバックログとリリース計画を維持する方法を見つける手助けをすることで、プロダクトオーナーの有効性を高めます。
（忘れずに！：プロダクトオーナーは優先順位付けがされたバックログの最終責任者です。）

□ プロダクトバックログは、プロダクトオーナーの現時点での考えの下に優先順位付けされていますか？
□ すべてのステークホルダーからの要求や要望はプロダクトバックログに反映されていますか？（忘れずに！：バックログは、創発的なものです。様々な人の意見に

よって相互作用が生まれ、個々では思いもつかないようなアイディアがもたらされます。）
□ プロダクトバックログは管理しやすいサイズですか？
多くのアイテムを管理維持するために、詳細になっているものを一番上に、大きなエピックのような（情報が少なく、曖昧な）アイテムは下におきます。
ただし、プロダクトバックログアイテムの優先順位が一番上である理由を、過去までさかのぼって過剰に分析するのは逆効果です。あなたが立てた仮説や戦略は、開発チームとステークホルダー／顧客間での継続的な対話によって変更される可能性があるからです。

□ 要求（特に、バックログの上にあるもの）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。

□あなたはこれまで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」について必要な情報を伝えていますか？ 自動テストとリファクタリングをDoneの定義とし、各アイテム毎に行えるようにすることが、難しい課題を紐解く一つの鍵となるかもしれません。

□バックログは情報発信の起点になっていますか？また、すべてのステークホルダーがすぐに理解できる状態になっていますか？

□バックログ管理に自動化されたツールを使用している場合、誰でも簡単に利用できますか？ 自動化された管理ツール（ソフトウェアなど）は、スクラムマスターからの積極的な情報発信のサポートがないと”情報冷蔵庫”になる危険性があるため、注意してください。

□あなたは情報をプリントアウトして皆に見せるなどして、情報発信を手助けしていますか？

□あなたは大きくて見やすいチャートを作成することで情報発信を手助けしていますか？

□あなたはこれまで、プロダクトオーナーが適切なタイミングでリリースしたり、優先順位をつけるためにバックログアイテムを調整するのを手助けしたことはありますか？

□リリース計画が最新の状態を保てているかどうか、皆知っていますか？
スプリントレビュー (※2)ミーティングでアイテムが「完了」と認識された後に、製品/リリースバーンダウンチャートを更新してください。実際に完了したPBIと新し

く追加されたPBIの両方の割合をグラフにすることで、範囲/スケジュールのずれを早期に発見できます。
□前回のスプリントレビューミーティングの後、プロダクトオーナーはリリース計画を調整しましたか？
十分にテストされた製品を予定通りに出荷することができるプロダクトオーナーは少数派ですが、彼らはスプリントごとにリリースを再計画します。なぜならスプリントレビューにより、優先されるべき重要な業務が明らかになるので、いくつかの作業を延期することが求められるからです。

Part II --開発チームの状況はどうでしょうか？

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リスクがあります。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” には以下のような特徴があります。(※3)

- 明確な目標（明確さ：期待値／必達が識別可能であること、達成可能であること、目標が自分 のスキルセットと能力に対して適切であること）
- 集中と選択ができ、範囲を特定した上で集中力を高く保つことを実践できる
- 自分への意識（頭で考えること）がなくなる感覚があり、行動と意識の融合が起こる
- 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る）
- 能力レベルと挑戦のバランス（簡単すぎることも、難しすぎることもない難易度）
- 状況や活動に対して、自己統制ができている感覚
- 本質的にやりがいがあるものなので、活動は無理のない形になっている

□ チームメンバーはお互いが好きで、ふざけ合ったり、お互いの成功を祝っていそうですか？

□ 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？

□ ひどく居心地が悪く、議論することを避けているような時や、そのような課題はありますか？(※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意識し続けていますか？ もしかすると、このスプリントでやると決めたプロダクトバックログアイテムの受け入れ条件（スコープ）を再確認するための、スプリント中間チェック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には、チームが実際に行っていることをどこまで反映していますか？ 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(※6)」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、出荷可能な製品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作成物（artifacts）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？

□ これらの作成物は、チーム外のお節介な人々から守られていますか？
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発的に業務を行っていますか？

□ 技術的負債を返済する必要性が”Done”の定義に明示的にされており、徐々にコードを書いていくことがより楽しく心地よいものになっていますか？

□ チームメンバーは、自分の職務範囲にとらわれず、合意した業務のすべての側面（例えばテスト、ユーザードキュメントなど）に対して責任を持って達成しようとしていますか？

Part III -- 開発プロセスはどうなってますか?

□ あなたたちの開発しているシステムには、回帰テストの失敗（以前動いていた機能が壊れてしまう）を見抜きやすい環境はありますか？
通常、これはxUnitフレー ムワーク（JUnit、NUnitなど）や、E2Eテスト（Cucumberなど）を活用して見抜きやすい環境づくりを行います。

□ 自動エンドツーエンドシステムテスト（機能テストなど）と自動単体テストは、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、グレーな領域を発見しましたか？(※7)

□ 回帰テストが失敗した時、CI(継続的インテグレーション)(※8)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.（日に１度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いますか？

□ 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(※9)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は難しいです。

□ "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、各PBI毎に実現できるようになっていますか？ 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。

□ チームメンバーは開発時間の多くをペアプログラミングに充てられていますか？ 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。 ペアで業務を行うことは、自身の作業方法や知見の見直しにも役立ちます。プライドが高かったり、改善を恐れる人にとっては、挑戦となるでしょう。最初は時間がかかるように見えますが、少しずつ業務に取り込みながら進めることで、ペアワークを好むようになるでしょう。

Part IV — 組織の状況はどうでしょうか？

□ チーム間でコミュニケーションは十分に起こっていますか？ 「Scrum of Scrums」はこれに近づくための一つの方法ですが、最善の方法ではありません。(※10)

□ チーム内で動くソフトウェア（顧客が判断できる結果）を作ることができますか？ 技術領域をまたいだチームになっていますか？(※11)

□ 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）」(※12)

□ あなたの組織は、チーム全体の目標と、組織の人事評価基準が二軸の両輪として存在していますか？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(※13)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、働くのに最高な場所、あるいは素晴らしいリーダーがいると、業界紙や独立機関などに認知されていますか？

□ “学習する組織”を創造していますか？

結論
上記項目にほとんどのチェックがつけられても、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ください。 人間の創意工夫を生み出す決まった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。
一度あなたがチームや組織に変化を起こすために、何が出来るかを考え始めると、実行に移すのが怖くなってくるかもしれませ ん。その気持ちは、あなたがスクラムマスターとして正しい道を歩んでいる証拠なのです。

※1 https://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).
※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).
※6　https://ja.wikipedia.org/wiki/暗黒物質
暗黒物質とは：質量は持つが、光学的に直接観測できないもののこと。
例えば、「テストを行った」と他のメンバーは言っていても、実際に中身を確認して皆が「確かに行っている」と認識するわけではないので、実際のテスト範囲や書

き方について過不足があるかどうか明らかではない状態の比喩として書かれています。
※7 グレーなエリアとは：ユニットテストとシステムテストの担当者が別々にいる場合（例えば、ユニットテストは開発チームが行い、システムテストはQA部が行うなど）、ソフトウェアコードの問題を見つけることは難しくなります。なぜならテスト範囲等はお互いに”任せてしまっている”からです。テストの範囲や内容について双方で理解しあわない限りは、「本当に期待通りにソフトウェアが動いているかどうか。問題はどこにあるか。」について知ることはできません。
※8 http://www.martinfowler.com/articles/continuousIntegration.html
※9 Martin Fowler, 「リファクタリング—既存のコードを安全に改善する—」(1999)
※10 http://less.works/less/framework/coordination-and-integration.html
を見て、他の方法を確認しましょう。
※11 http://FeatureTeamPrimer.org/
※12 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)
※13 Alfie Kohn, 「報酬主義をこえて」 (1999)