スクラムマスターチェックリスト

MJ (Michael James)
mj@seattlescrum.com
2022年2月8日改訂

https://scrummasterchecklist.org
翻訳：清水 弘毅(しみず こうき)

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。おそらく致命的なこと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、一度に一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 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のように過ごしていますか？

スクラムマスターは、様々な方法でプロダクトオーナーの有効性を高めます。

☐ チーム数に関係なく、プロダクトに対してプロダクトオーナーは一人（だけ[1]）ですか？

☐ 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で優先順位付けされていますか？

☐ 新しく発見された要求や要望は、プロダクトバックログに取り込まれていますか？プロダクトバックログは”創発的”であることを忘れないでください。

☐ プロダクトバックログは管理しやすいサイズですか？管理可能なプロダクトバックログアイテム数を維持するためには、優先順位が高いほど粒度を細かくし、優先順位が低いほどエピックと呼ばれる大きく一般的なエピソードが入ります。また、優先順位が低いプロダクトバックログアイテムを詳細に分析しすぎるのは逆効果です。計画は、開発中のプロダクトとユーザー/顧客との継続的な会話の中で変化していくものです。

[1] https://scrummaster.jp/why-scrum-isnt-making-your-company-very-agile/

☐ プロダクトバックログの（特に）優先順位が高いアイテムは、独立した、交渉可能な、価値のある、見積もり可能な、小規模な、そしてテスト可能な（INVESTな）ユーザーストーリー[2]として表現できていますか？

☐ 技術的負債とそれを回避する方法について、プロダクトオーナーを教育したことがありますか？自動テストとリファクタリングを、各プロダクトバックログアイテムの完成(Done)の定義に書き込むことが、難しい問題を解く一つの鍵になるかもしれません。

☐ プロダクトバックログは情報発信の起点であり、スクラムチーム、チーム外のステークホルダーを含めて関係者全員がすぐに見ることができる状態になっていますか？

☐ プロダクトバックログの管理に自動化されたデジタルツールを使っている場合、誰もがその使い方を簡単に把握できているでしょうか？悲しいかな、自動化されたデジタルツールはたいてい、チームの協調を阻害する情報冷蔵庫になってしまいます。

☐ 大きくて見やすいチャートの作り方を教えることで、情報発信を手助けできていますか？

☐ 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プロダクトバックログアイテムを調整するのを支援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、スクラムチームやステークホルダーは把握していますか？スプリントレビューミーティングでプロダクトバックログアイテムが「完了」とされた後に、プロダクトまたはリリースバーンダウンチャート[3]を更新して、全員に見せるのはどうでしょう？完了したPBIと新たに追加されたPBIの割合をグラフにすることで、スケジュールとスコープのずれを早期に発見できるかもしれません。

☐ スプリントレビューの後に、プロダクトオーナーはリリース予測を調整しましたか？もし、十分にテストされたプロダクトを決まった日までに出荷したいのであれば、常にスコープを考え直す必要があります。そのため、より重要な作業が途中で発見された場合、一部のプロダクトバックログアイテムを将来のリリースに延期する必要があるかもしれません。 プロダクトによっては、スプリントごとにリリースすることもできますし、スプリントの中で複数回リリースすることもできます。

Part II --開発チームの状況はどうでしょうか？

スクラムマスターは、チームメンバーと協力して仕事を進めるという模範を示すことが求められますが、技術的なタスクに没頭してしまう危険性があります。チームに対する主な責任について考えてみましょう。

☐ あなたのチームはフロー状態[4]になっていますか？この状態の特徴は以下の通りです。:

- 明確な目標（期待やルールが明確であり、自分のスキルや能力に合った達成可能な目標である）
- 選択と集中ができ、限られたこと・もの・場所に注意を向けることができる状態

---

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

[3] Mike Cohn, *Agile Estimation and Planning*. (2005).

[4] Mihaly Csikszentmihalyi, *Flow: The Psychology of Optimal Experience* (1990).

- 迷いがなく、一心不乱な状態。
- 直接かつ即時的なフィードバックを受けれる状態（活動の過程での成功や失敗が明らかで、必要に応じて行動を適応することができる）
- 現在の能力レベルとチャレンジのバランスが適切な状態。（アクティビティは簡単すぎず、難しすぎず）
- 状況や活動を自分自身で選択し制御できているという感覚がある状態。
- 本質的にやりがいのある活動であり、楽しさがある状態

☐チームメンバーはお互い気が置けない仲で、一緒にふざけたり、お互いに成功を祝ったりするような状態ですか？

☐ チームメンバーはお互いに高い責任を持ち、成長するために挑戦し合っていますか？

☐ 本来話合うべきなのに、チームが背を向けて、話し合われていない問題はないでしょうか？[5]

☐ スプリントレトロスペクティブのいろいろな手法や（気分転換できる）開催場所等を試しましたか？[6]

☐チームはスプリントゴールに集中し続けていますか？スプリント計画を見直すための、スプリント中間確認ポイントを設けてもいいかもしれません。

☐ スプリントバックログは、チームが実際に行う予定のもの、行ったものを反映されていますか？誰も知らない未公開のタスクや1日分の仕事より大きなタスクなど、「ダークマター[7]」に注意しましょう。 スプリントに関係のない仕事は、スプリントの妨げになります。

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

☐スプリントバックログは最新の状態になっていますか？

☐ チームが自己管理している作成物(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ものになっていますか？

☐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（ボランティアのように）仕事をしていますか？私が担当した最初のスクラムチームは、まるでお給料をもらっているボランティアのような感じがました。あなたのチームにそうした感じがないのであれば、何かが誤っているかもしれません。

---

[5] マーシャル・B・ローゼンバーグ, *NVC* 人と人との関係にいのちを吹き込む法 新版 (2018).

[6] 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 (2007).

[7] 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。

☐ 技術的負債返済の必要性がdoneの定義に明示され、コードを書くことがより楽しく心地よい感じになっていますか？

☐ 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（テスト、ユーザードキュメントなど）のすべての側面にチームとして責任を持って取り組んでいますか？

Part III — エンジニアリングプラクティスはどうなっていますか？

☐ 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）と自動化されたユニットテストのバランスは適切ですか？

☐ チームは、システムテストとユニットテストの両方を、開発中のシステムと同じプログラミング言語で書いていますか？チームの一部メンバーだけがメンテナンス方法を知っている独自のスクリプト言語やキャプチャ再生ツールでは、チーム内の協働は促進されません。

☐ あなたのチームは、システムテストとユニットテストの間の有用なグレーゾーン[8]を発見しましたか？

☐ 回帰テストが失敗した時、CI(継続的インテグレーション)[9]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）

☐ すべてのテスト結果をCIサーバに上げていますか？

☐ チームメンバーは、Big Up Front Designに代わる、継続的な設計と絶え間ないリファクタリングの楽しさを発見しましたか。リファクタリングには厳密な定義があり、「外部の動作を変えずに内部の構造を変更すること」です。リファクタリングは継続的に、あるいは少なくとも1時間に数回行う必要があります。 重複したコード、複雑な条件ロジック（過剰なインデントや長いメソッドでわかる）、不適切な名前の識別子、オブジェクト間の過剰な結合などがあれば、いつでもリファクタリングします。自信を持ってリファクタリングするためには、自動化されたテストカバレッジがあればこそです。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、悪いコードに取り組んでくれる優秀な開発者を見つけることは難しいからです。

☐ "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。

---

[8] グレーゾーンとは：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。

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

☐ 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やモブ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、 欠陥率を減らすことができます。 この手法は、チームメンバーの境界に挑戦し、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、コード行数で測定した場合）。スクラムマスターは、ペアワークやモブセッションを率先して始め、模範を示しましょう。そのうちこの手法で仕事をすることを好む人も出てくるでしょう。

Part IV — 組織の状況はどうでしょうか？

☐ チーム間でのコミュニケーションは適切に行われていますか？"スクラム・オブ・スクラム “は、これを実現するための一つの方法に過ぎず、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）を作ることができていますか？アーキテクチャや技術領域をまたいだチームになっていますか？[11]

☐ 組織内の複数チームが、横断的、組織的、システム的な問題を解決するために、全体ふりかえり（オーバーオールレトロスペクティブ）を実施していますか？*[12]*

☐ 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、透明性が高い状態になっていますか？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）[13]

☐ 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[14]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。

---

[10] https://scrummaster.jp/seven-alternatives-to-scrum-of-scrums-jp/

[11] https://featureteams.org/jp/feature_team_primer_ja.pdf

[12] https://less.works/less/framework/overall-retrospective?preferred_lang=jp

[13] ケン・シュエイバー, スクラム入門-アジャイルプロジェクトマネジメント (2004)

[14] Alfie Kohn, *Punished By Rewards: The Trouble with Gold Stars, Incentive Plans, A's, Praise, and Other Bribes* (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:

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。