

サイバーエージェント アドテクスタジオ 茅野 祥子

**ただのエンジニアが
チームの改善を謀ったら。。。。**

茅野 祥子(@shokotachocota)



- ▶ サーバサイドエンジニア
- ▶ エンジニアとしての歴史
 - ▶ 2010年 web企業に新卒入社
 - ▶ 2014年 CyberAgent adtechstudio Dynalyst 所属
- ▶ アジャイルとの関わり
 - ▶ 2011 アジャイルサムライを初めて読む
 - ▶ 2014 転職後、アドテクスタジオのゼミ制度でゼミに参加
 - ▶ 2015 CSM / CSPO



リーン・アジャイルゼミ

▶ ゼミ長：横道 稔

▶ 『あなたのチームの「いい人」は機能していますか？』

▶ podcast : omoiyari.fm



スクラムできてますか

ちゃんとしたスクラムって難しいんですよ

- ▶ 本来のスクラムは、守破離の精神にのっとり、まず「守」、フレームワーク通りの実行をすべき
- ▶ 実際はすでにチームでのやり方が確立していたり、メンバーの変動があったり、事業の事情だったり難しい
- ▶ スクラム自体に対しての理解や認識はあるので一部取り入れることに対しては積極的なメンバーが多い
- ▶ 結果、「スクラムっぽいこと」をしているチームが多数
- ▶ スクラムマスターがいなくても成り立つ世界

スクラムっぽいことが成り立つ世界

- ▶ 人数：5~8程度
- ▶ ある程度成熟してコミュニケーションの取れるメンバー
- ▶ 各ストーリーに対して完了に向けて協力できる状況
- ▶ 個人が全体を把握できる規模
- ▶ 割り込みが少ない
- ▶ 事業の目線とメンバーの目線がずれていない

- ▶ ある程度のスクラムの要素である程度健全に開発が回る世界
- ▶ スクラムマスターがない（数名それっぽい人がいる）で成り立つ世界

スクラムマスターって何
ですか？

スクラムマスターとは

スクラムフレームワークが正しく適用されていることを保証する役割であるが、権限としては間接的である。主な作業は、チーム内外の組織間調停(ファシリテーション)と外部妨害を対処することとされる。従来のプロジェクトマネージャがこの役割を担うことが多いが、プロジェクトそのものを管理するわけではない。

from wikipedia



The screenshot shows the Japanese Wikipedia page for "スクラム (ソフトウェア開発)". The page title is "スクラム (ソフトウェア開発)". The main text describes Scrum as a software development methodology. It mentions that Scrum is a type of agile software development methodology. The text states that Scrum is a methodology for software development that is iterative and incremental. It emphasizes that Scrum is a framework that allows teams to manage and complete their projects by applying it to their work. The text also notes that Scrum is a framework that allows teams to manage and complete their projects by applying it to their work. The page includes a search bar at the top right and a navigation menu at the top left. The page is in Japanese and is part of the Japanese Wikipedia project.

スクラムマスターとは

- ▶ 開発手法は別にスクラム以外にもあるが、スクラムマスターという役割は必要である
- ▶ スクラムマスターは、開発チームの **医者** である
- ▶ 開発チームが健全であるために、守るための存在

事業が拡大して見え
てきたこと

事業の拡大におけるスクラムの課題

- ▶ それまでと同じ「スクラムっぽいこと」では健全ない世界
- ▶ エンジニアの増加
- ▶ ビジネスメンバーの増加
 - ▶ 運用の肥大化
 - ▶ 割り込みの増加
- ▶ プロダクトの仕様の複雑化、機能の増加

ビジネスメンバー（チーム外メンバー）の増加

▶ 課題

- ▶ 外部からの相談、割り込みが増える
 - ▶ 開発に集中できない状況
- ▶ 運用コストの増加
 - ▶ 一部メンバーが運用コストで疲弊する
 - ▶ 運用でストーリーが進まない
 - ▶ 運用改善のストーリーの優先度

▶ 対策方法

- ▶ 窓口のメンバー固定する（相談、運用） = スクラムマスター的なポジション

プロダクトの仕様の複雑化、機能の増加

▶ 課題

- ▶ クライアントからの要望、市場の拡大による機能追加
- ▶ チームを跨いだ仕様の共有が必要になる

▶ 対策

- ▶ プロダクトマネージャーの立場を確立させる
 - ▶ プロダクトマネージャーがプロダクトマネージメントできるようにする
- ▶ チーム全体のスプリント計画MTG

エンジニアの増加

▶ 課題

- ▶ 共有・相談の難しさ、朝会の満足度
- ▶ どこで誰がなにをしているかわからない、把握できない
- ▶ 知らないことが増えることでプロダクトに対するモチベーションの低下
- ▶ 責任をとる人が偏りがち

▶ 対策方法

- ▶ チームを分ける（5~6名程度）
- ▶ LeSSの導入

LeSS BOOK CHAPTER 2:
INTRODUCTION



LEAN THINKING



PRINCIPLES



SYSTEMS THINKING



COACHING



ADOPTION



CONTINUOUS IMPROVEMENT

FEATURE TEAMS



TEAMS

STRUCTURE



ORGANIZATION



COMMUNITIES



TECHNICAL EXCELLENCE



CONTINUOUS INTEGRATION



ARCHITECTURE & DESIGN



ROLE OF MANAGERS

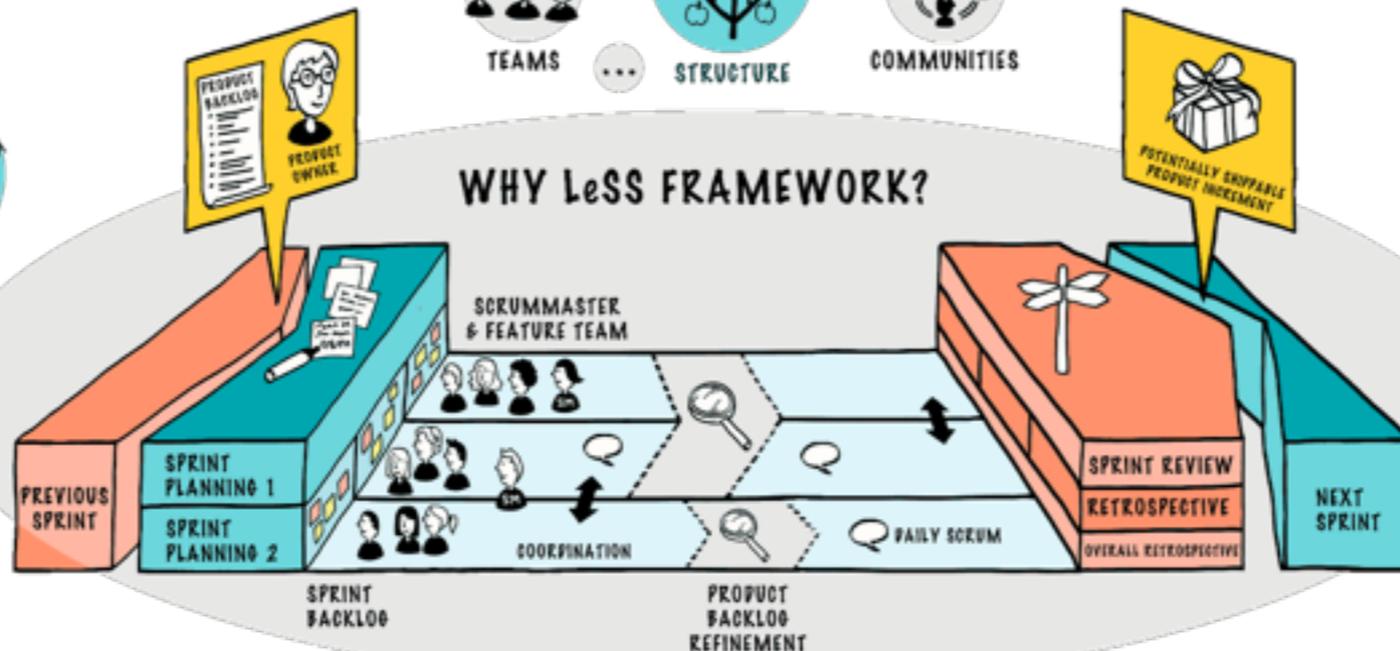


MANAGEMENT



GO SEE

WHY LeSS FRAMEWORK?



PRODUCT OWNER TEAM



LeSS HUGE



REQUIREMENT AREAS

治療が必要



スクラムマスターとい
うポジションの難しさ

エンジニアから見たスクラムマスター

- ▶ 邪魔なことを除外してくれる人
- ▶ 開発以外の課題を解決してくれる人
- ▶ 働きやすくしてくれる、開発に集注する手助けをしてくれる人
- ▶ 困った時に助けてくれる人

- ▶ いなくても開発はできると思われがち
- ▶ 困ったらエンジニア自信が解決するために働きかければよい
 - ▶ 変なストレスがたまる
 - ▶ 開発より優先度が上がらずいつまでもズルズルと解決しない

プロダクトマネージャーから見たスクラムマスター

- ▶ 開発をスケジュール通りに進めてくれる人
- ▶ 開発タスクやバックログの内容について聞けば答えてくれる人
- ▶ 技術的な内容を教えてくれる人
- ▶ 困った時に助けてくれる人

- ▶ いなければ近くのエンジニアに相談すればいい
- ▶ 「自分がやるよ!!!」
 - ▶ プロダクトを考えることに集中できない

スクラムマスターというポジション、採用の難しさ

- ▶ エンジニア、プロダクトマネージャから見ても、スクラムマスターの重要性が伝わりづらい
- ▶ エンジニアさえ枯渇している今の時代、スクラムマスターを雇う余裕があるなら、その分優秀なエンジニアを雇いたい
- ▶ ちゃんとしたスクラムやってないし、フレームワーク通りにやらないとダメなんですよ
- ▶ 結果、不健全な開発環境の中でエンジニアだけ増えていき、一人一人のパフォーマンスは下がり、コストだけかかる負のスパイラルが発生する
- ▶ メンテナンスされないバックログ、カンバンが出来上がりがち

スクラムマスター専属
のポジションを置くこと
は難しい

エンジニアがスクラムマ
スターをしてみた

エンジニアがチームの治療をするために

- ▶ 「スクラムちゃんとやりましょう」とは言わない
- ▶ あくまでも1チームメンバーとして、スクラムの小さな改善を進める
- ▶ 開発と同じくらい、チームメンバーに目を向ける
- ▶ エンジニアリング以外のタスクを積極的にとる
 - ▶ 「いい人」に自分から積極的にになった
 - ▶ エンジニアとしてすべき仕事は他のメンバーにお願いした
- ▶ 健全なスクラムになるように裏から促すように動いてみた
- ▶ メンバーが100%の力を出せる環境を作るように心掛けた

チャレンジできる環境があった

- ▶ ゼミの存在
 - ▶ 私がスクラムに興味をもって勉強してるという認識があった
 - ▶ ゼミメンバーに相談ができた
- ▶ スクラムに「嫌悪感」をもったメンバーがいなかった
 - ▶ やってみたいことに対して否定的なメンバーはいなかった
- ▶ スクラムにすごく興味をもったメンバーがいなかった
 - ▶ 勝手にいろいろ試せた

例1. スプリントレトロスペクティブ

▶ 課題

- ▶ それまでのスプリントの終わりは、今週終わったこと、終わらなかったこと、来週やること、と進捗の確認しかできていなかった。
- ▶ 解決作：ふりかえりを提案してみる。
 - ▶ なぜ終わらなかったのか。
 - ▶ 終わらせるためにはどうすればよかったのか。
 - ▶ 次はこういう改善をしてみましょう。
- ▶ ※ 「レトロスペクティブ」という単語は絶対出さない

例2. 外部妨害の排除

▶ 課題

- ▶ 各エンジニアで属人化している運用や、差し込みタスクがあり、見積もりより開発に時間がかかっていた。

▶ 解決策

- ▶ 各エンジニアが持つ運用を自分に集約してみた。
- ▶ 外部には、エンジニアへの依頼、相談は自分にするように集約してみた。

エンジニアとスクラムマスターの兼務

▶ メリット

- ▶ システムの理解と、解決手段をある程度判断できるため、外部とのコミュニケーションに支障が起きづらい
- ▶ 外部妨害に対する対処として、ちょっとした改修や修正を自分ですることができる
- ▶ 開発メンバーとのコミュニケーションが取りやすい

▶ デメリット

- ▶ バックログ上のストーリーに参加すると自分が足を引っ張りがち
- ▶ ベロシティが計りづらい
- ▶ オーバーワーク

エンジニアとしてスクラムマスターをしてみて…

- ▶ エンジニアとしての開発業務は減った
 - ▶ 元々開発プロセスに関して興味があったのでモチベーションはそこまでさがらなかった
- ▶ 評価してもらえない状況は起きていない
 - ▶ あくまでも1エンジニアなので、評価してくれるのはエンジニアの上長であり、エンジニアの上長は、私のスクラムマスター的な仕事をみて判断をしてくれていた。
- ▶ ハードワークさは否めない
 - ▶ メインストーリーの開発などは他のメンバーに任せ、自分はサブタスクなどポオトルネックにならないタスクを取るようにした

まとめ

- ▶ フレームワーク通りのスクラムでなくてもスクラムマスターは必要
- ▶ スクラムマスターは、チームの医者であり、チームを健全にするために守る存在
- ▶ エンジニアがスクラムマスターを兼務してみると、スムーズにスクラムマスターのポジションを確立することができた

ご清聴ありがとうございました