# **Teams**

チーム

## What is a team?

チームとは？

Teams are the core building block for large product development—and team structure has a huge impact on productivity and cycle time.

チームとは大規模開発において中核をなす要素であり、チーム構造は生産性とサイクルタイムに絶大なインパクトをもたらす。

Team has:

チームは以下を有す

* a shared work product

作業製品の共有

* interdependent work

相互依存の仕事（「作業」の方がしっくりくれば）

* a shared responsibility

共同責任

* a set of working agreements

労働協定のセット

* responsibility for managing the outside-the-team relationships [[SJS03]](http://www.amazon.com/The-Team-Handbook-Third-Edition/dp/1884731260)

チーム外との関係性を管理する責任

→[Link]The Team Handbook Third Edition - Amazon.com

* distributed leadership [[Katzenbach98]](http://www.amazon.com/Teams-At-Top-Jon-Katzenbach/dp/0875847897)

分散型リーダーシップ

→[Link]Teams At the Top - Amazon.com

## Self-managing teams

自己管理チーム

Self-managing teams are the basis of Scrum and a widespread modern management practice. The team has the authority to design, plan, and execute their task and to monitor and manage their work process and progress[[Hackman02]](http://www.amazon.com/Leading-Teams-Setting-Stage-Performances/dp/1578513332). The team rather than a (project) manager—has the responsibility of deciding how to work.

自己管理チームはスクラムの基礎であり、近代的な管理慣習として広く知られている。チームはタスクをデザイン・計画・実行し、また業務の進行状況・進捗をモニターし管理する権限を持つ。（プロジェクト)マネージャーではなく、チームそのものが、どのように働くかを決める責任を持っている。

→[Link]Leading Teams: Setting the Stage for Great Performances - Amazon.com

Self-managing teams do not just happen, they need the right environment. The organization is responsible for supporting the team development by creating the conditions needed for teams to succeed. Switching to self-managing teams means the job of the traditional manager changes from directing the team to creating these conditions.

自己管理されたチームは自然発生しない。その誕生には常に適した環境が必要である。組織はチームが成功を収めるのに必要な環境を整えることで、チーム形成をサポートする責任がある。自己管理されたチームへの転換とは、マネージャーの役割が従来型のチーム管理から、これらの環境を整える作業への変換を意味する。

## Cross-Functional Teams

相互協力チーム

Self-managing teams are cross-functional (or multifunctional). A cross-functional team is a “*group of people with a clear purpose representing a variety of functions or disciplines in the organization whose combined efforts are necessary for achieving the team’s purpose*” [[Parker02]](http://www.amazon.com/Cross--Functional-Teams-Working-Strangers/dp/0787960853).

自己管理されたチームは相互協力できるチームでもある(もしくは多機能型チームとも)。相互協力できるチームとは、組織の中で様々な機能や教練を発揮し、チームの目的達成のために協力が必要である、明白な目的を持った人々の集まりである。

→[Link]Cross- Functional Teams: Working with Allies, Enemies, and Other Strangers - Amazon.com

By definition, a Scrum team is cross-functional. Its members include at least product marketing (Product Owner), software development, and testing. A cross-functional team implies breaking the organizational barriers between development and testing by putting them in the same team

Cross-functional means that team membership includes all the key functions involved in the project, usually Engineering, Marketing, and manufacturing, at a minimum [[Smith07]](http://www.amazon.com/Flexible-Product-Development-Building-Changing/dp/0787995843).

スクラムチームは相互協力できるチームと定義できる。そのメンバーには少なくとも製品マーケティング（プロダクトオーナー）、ソフトウェア開発、及びテスター（試験者）を含む。相互協力できるチームには、開発者とテスターを同じチーム内に配置することによる、組織内の障壁破壊を期待している。

相互協力とは、プロジェクトに関係する全てのキー要素を含むチームメンバーシップを有することを意味し、一般的には少なくともエンジニアリング、マーケティング、および製造を含む。。

→[Link]Flexible Product Development: Building Agility for Changing Markets - Amazon.com

## Team based organization

チームは組織の基礎

Team-based—LeSS—organization has the following structure:

チームによって構成されたLeSS組織は以下の構造をもつ。

* **Dedicated teams**—Each team member is dedicated for 100% of his time to one and only one Team. This might feel inflexible, but team members require dedication if you want them to (1) take a shared responsibility for the Team’s goal, and (2) take ownership of how a team works—own their processes.

専属チーム ― 個々のチームメンバーはひとつだけのチームに所属し、これに100%専念する。これは応用力に欠けると感じるかもしれないが、チームメンバーが①チームのゴールを達成するための責任を共有し、 ②チームの働き方に対するオーナーシップを持つには、専属制が必要となる。

* **Cross-functional teams**—Each team contains all functional skills needed to produce a shippable product. Traditional functional specialization might feel the most ‘efficient’ from that function’s perspective, but most effort and problems in product development is “between the functions” and thus teams require to be cross-functional if you want them to focus on the whole working product.

相互協力(機能横断)できるチーム ― 個々のチームが、出荷できる製品を生産するために必要な全ての機能的なスキルを有しす。あるひとつの機能の視点から見れば、機能性に特化した伝統的な手法が最も効率的であるかのように映るが、製品開発における多くの問題や費やされる労力は「機能間」に生じるため、チームが生産の全体像にフォーカスする上で、相互協力が必要となる。。

* **Co-located teams**—Each team is co-located in the same room. This might sound unreasonable. Wouldn’t you, in today’s globalized world, want to use the best skilled people in the place where they are? No. We want the best teams that can take a shared responsibility for the outcome of the Team, and learns from each other. Shared responsibility requires trust. Humans build up trust quickest be close cooperation and face-to-face communication. Co-location also promotes team learning—the essense of continuous improvement.

チーム全員集合 ― 個々のチームは同じ部屋に配置される。これは時として理不尽に聞こえるかもしれない。今日のような国際化された社会では、能力の長けた人材をを本来の配置で使いたいと思うであろう。しかし、我々はチームの成果のために責任を共有でき、そして互いに学びあえるベストなチームがほしいのだ。責任の共有には信頼を必要とする。人間が信頼関係を構築する最も迅速な方法は、密接な協力と対面コミュニケーションだ。同じ場所に配置されることによりチームは学習し、これは継続的な改善の要素として働く。

* **Long-lived teams**—A Team stays together ‘forever.’ This might feel idealistic, but Teams need to have stability if you want the team to care about how they work as a Team. Anyone who has ever been on a real long-lived team knows that teams get better as the team members get to know each other and learn how to do and improve work together.

長命なチーム ― チームは「永久に」協同する。これは時として理想論と感じるかもしれないが彼らがチームとしてどのように働くかを気にしていてほしいのであれば、チームには安定性が必要となる。これまで長命なチームにいた経験を持つ誰もが、チームメンバーがお互いにを知るようになり、共に働き、また改善する方法を学ぶことで、チームがより良くなることを知っている。

## Team manages external dependencies

外的関係を管理する

What are the implication for large product development? They are profound! First, each team needs a clear goal so that they know their boundaries. Establishing customer-focused feature teams helps. It results in code-focused, cross-team communication. Second, the organization needs to make it crystal clear that the teams themselves are responsible for coordinating their work with other teams. A team’s success must be measured by the whole product’s success to prevent local optimization. Removing official coordination roles such as project managers makes it clear to the team that coordination is their responsibility. Third, establish a whole product-wide continuous integration system. This creates the visibility teams need so that they can coordinate their work. The health of the product must always be visible to everybody.

大規模開発による影響とは何か？ それはとても深い！まず、個々のチームは作業領域を把握するため、明確なゴールが必要だ。顧客視点を確立はチームの助けとして機能し、結果としてコードに着目した、チーム間コミュニケーションが形成される。次に、組織はチーム自身が他のチームとの調整を行う責任をもつ点について、明確にする必要がある。チームの成功は個別最適化を防ぎ、製品全体の成功によって評価されなければならない。プロジェクトマネージャのような公な調整役を取り除くことで、チーム自体が調整に責任を持つことを明確にする。3つ目に、製品全体の継続的なインテグレーションシステムを構築する。それらを構築することでチームの要件を可視化し作業調整が可能になる。製品の健全性は誰の目にも明らかでなければならない。

## Multi-skilled Workers

多様なスキルを持つ作業者

Learning is the major activity in product development. In the long run, reducing learning reduces efficiency—not increases it. Sherman, in Fortune Magazine:

製品開発において学習とは重要な要素である。長い目で見れば、学習を縮小することは効率化を下げ、増大させることはない。フォーチューン誌でシャーマンは次のように述べている；

Workers will be rewarded for knowledge and adaptability. Specialization is out, a new-style generalism is in. The most employable people will be flexible folk who can move easily from one function to another, integrating diverse disciplines and perspectives… people will need the ability not only to learn fundamentally new skills but also to unlearn outdated ways.

作業者はその知識と適応性が評価される。、全体主義という新しいスタイルが、専門性に取って代わる。最も雇用に適した人々は、1つの機能から別のものに容易に動くことができる柔軟な人々になるだろう。多様な分野と視点を統合する……根本的に新しいスキルを学ぶだけではなく、旧式な方法を忘れる能力が必要である。

## Team makes decisions

チームが判断する

Self-managing teams make their own decisions. However, many people grew up in a command-and-control environment where management made decisions for them. A ScrumMaster can help the team learn how to make decisions. A team agreement on how to make decisions is more important than the specific decision-making method.

自己管理チームは自ら判断を行うことが出来る。しかし、多くの人々は管理者が判断を行う命令統制下で成長してきた。チームにおいて判断を行う手法の学習に、スクラムマスターが一役を担うどのように判断を行うかに関する合意形成は、判断を行う特定の手法より重要である。

The Decider protocol [[MM02]](http://www.amazon.com/Software-Your-Head-Protocols-Maintaining/dp/0201604566), is a quick and easy way of making consensus decisions.

決定者の手法は、合意形成をするための素早く容易な方法である。

→[Link]Software for Your Head: Core Protocols for Creating and Maintaining Shared Vision - Amazon.com

## Have conflict in the team

チーム内衝突

People working together creates conflict. That is not a bad thing. But conflict needs to be resolved. Unresolved conflict has a negative impact on team performance and creates a dysfunctional team atmosphere [Lencioni02]. Resolved conflict, on the other hand, creates learning and trust, both of which have a positive impact on performance. Conflict is an opportunity for the team to improve their performance, and hence a good thing.

人が一緒に働けば衝突が生じる。これは悪いことではない。しかし、衝突には解決が不可欠である。未解決の衝突はチームのパフォーマンスに悪い影響を与え、、チームの空気に機能障害をもたらす。一方で解決された衝突は、学習と信頼をもたらし、パフォーマンスに良い影響トを与える。衝突はチームにとってパフォーマンスを改善する好機と言える。ゆえに良い事と言えよう。

## Don’t have phase-based “resource allocation”

フェーズ別リソース割り当てをしない

Scrum is not the waterfall. There are no phases. With its self-managing, cross-functional, long-lived feature teams, it balances the “resource need” over the release. The same people stay on the release from the beginning until the end.

スクラムはウォーターフォールでは無い。フェーズは存在しない。自己管理、相互協力、長命といった特性を持つチームで、リリースに向け「必要な人的リソース」のバランスを測る。リリースに際して同じ人員がプロジェクトの最初から最後まで在籍する。

## Conclusion

結論

These different—but proven—team concepts cause major change in organizations.

これらの立証されたチームコンセプトの違いは、組織に大きなな変化を起こす。

* Self-managing teams require a change from command-and-control management to manager-teacher. Instead of focusing on what people do, management should focus on how to create the environment for the teams to succeed.

自己管理チームには命令統制下の管理から管理指導への変化を要求する。管理者は人とその行動にフォーカスするのではなくチームを成功に導くための環境の作り方にフォーカスするべきである。

* Cross-functional teams require breaking functional boundaries and working together across the whole organization to optimize delivering customer value. Instead of boxing people in functional groups, management should focus on cross-functional learning.

相互協力チームは効果的に顧客価値を届けるために、機能的な境界を破壊し、組織全てが共に働くことを必要とする。人々を形式的な機能別グループに割り振る代わりに、管理者は相互協力による学習に着目すべきである。

* Long-lived dedicated teams require giving work to existing teams and letting them decide how to do it. Instead of considering individuals to be the unit of performance, the focus needs to be on complete teams.

長命な専属チームは、既存のチームに仕事を与え、どのようにそれをこなすかを決めさせることを必要とする。個々の能力をパフォーマンスの単位として捉えるのではなく、チーム全体に注目しなければならない。

## Recommended Reading

推奨図書

When switching to cross-functional teams, changing management style is difficult. Luckily, a lot of excellent material has been written on this subject.

相互協力型チームに移行するために必要な管理体制の変換は困難である。以下に参考となる文献を紹介する。

* [Leading Teams, by Richard Hackman.](http://www.amazon.com/Leading-Teams-Setting-Stage-Performances/dp/1578513332)
* Harvard professor Richard Hackman is a long-time team researcher. His book is currently our favorite team-related book. It has a strong focus on helping management in their change to team-based work.
* ハーバード大教授のリチャード・ハックマン氏はベテランの研究者で、彼のチームに関する著書は現在我々の愛読書になっている。チームを基本とした仕事において管理者に求められる変革に焦点を宛てている。
* [Leading Self-Directed Work Teams, by Kimball Fisher.](http://www.amazon.com/Leading-Self-Directed-Teams-Kimball-Fisher/dp/0071349243)
* This book has a strong focus on the change in role when one becomes a team leader of a self-directed team.
* この書籍は、自立したチームにおけるリーダーの役割に注目している。
* [The Wisdom of Teams, by Jon Katzenbach and Douglas Smith.](http://www.amazon.com/Wisdom-Teams-High-Performance-Organization-Essentials/dp/0060522003)
* This is probably the most popular team reference and certainly worth reading.
* おそらく、最も一般的なチームの参考書として一読の価値がある。
* [The Five Dysfunctions of a Team, by Patrick Lencioni.](http://www.amazon.com/Five-Dysfunctions-Team-Leadership-Fable/dp/0787960756)
* Written like a novel, it covers well the need for conflict in teams.
* チーム内における衝突の必要性を小説調に記す。

Cross-functional teams are described mainly in product development literature. Some good texts:

相互協力チームは製品開発に関する書籍でよく取り扱われる。以下に数点を紹介する。

* [Fast Cycle Time, by Chris Meyer.](http://www.amazon.com/Fast-Cycle-Time-Strategy-Structure/dp/141657624X)
* This is a true classic on product development and talks about cross-functional (multifunctional) teams in detail.
* 製品開発における好例として、相互協力（多機能型）チームについて詳細に記されている。
* [Revolutionizing Product Development, by Steven Wheelwright and Kim Clark.](http://www.amazon.com/Revolutionizing-Product-Development-Quantum-Efficiency/dp/0029055156)
* Another classic in product development literature; has one chapter on cross-functional integration.
* 製品開発における好例として、相互協力の導入に関して一つの章を割いている。

Some texts related to software development teams:

ソフトウェア開発チームに関連する書籍

* [Software for Your Head, by Jim and Michele McCarthy.](http://www.amazon.com/Software-Your-Head-Protocols-Maintaining/dp/0201604566)
* Jim and Michele spent years in ‘boot camps’ to find the most efficient ways for teams to work. They documented this as a set of protocols in this book.
* ジムとミシェルは最も効率的なチームの運用を学ぶため、「ブートキャンプ」に歳月を費やした。その成果をプロトコルのセットとして執筆した一冊。
* [Peopleware, by Tom DeMarco and Tim Lister.](http://www.amazon.com/Peopleware-Productive-Projects-Teams-3rd/dp/0321934113)
* This classic on the importance of people in software development also has a couple of chapters focusing on teams.
* ソフトウェア開発における人材の重要性について示した本書の中には、チームに関する章が複数含まれている。