開発がわかる運用組織、内製できる運用組織

Internet Week 2023

運用設計ラボ合同会社

シニアアーキテクト 波田野 裕一

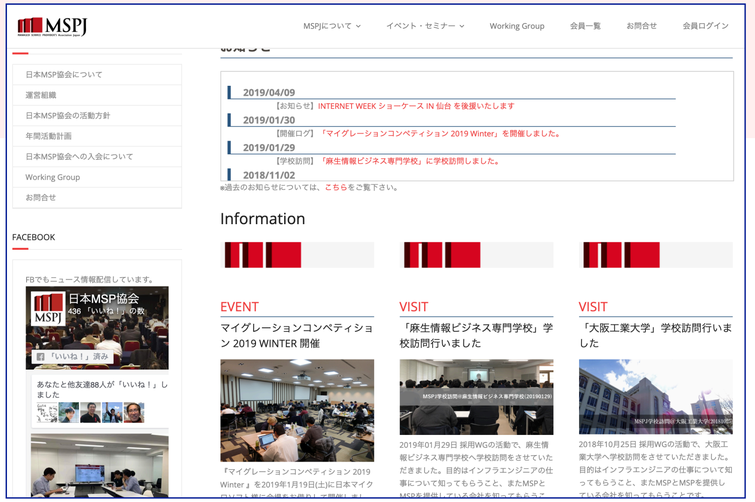
2023-11-20

日本MSP協会

IT情報基盤の運用サービスを提供するマネージド・サービス・プロバイダ及びIT情報基盤の運用に携わる技術者等と協力し、**運用の技術向上と品質向上、運用技術に携わる人材の発掘と育成、運用に関連する様々な評価軸を整理して明確化**するために日本MSP協会を設立します。そして、利用者にとって最適なIT情報基盤の選定と、適切なコストで安全かつ効率的に基盤を運用する指標を提供することで、さらなるIT産業界の活性化に貢献していきます。

MSPJ
MANAGED SERVICE PROVIDER'S Association Japan

https://mspj.jp/



運用設計ラボ合同会社

Operation 運用設計Lab

シニアアーキテクト

波田野 裕一

AWS Samurai 2017 (個人)
AWS Samurai 2020 (CLI専門支部)

AWS Community Hero

日本MSP協会 特別会員

インプットご支援

OpsLearn ®

科学的工学的な考え方に基づく講義とワークショップで
30年先も活きる運用設計スキルを身に付ける

OpsCLI ®

世界で最もAWS APIの仕様に忠実なeラーニングで
10年先も活きるAWSスキルを身に付ける

現場での実践ご支援

運用設計支援　　よろず相談

アウトプットご支援

ホワイトペーパー/ガイドライン策定支援
ホワイトペーパやガイドラインの制作や内製をご支援いたします。

概要

運用組織は、「開発と運用」という軸の一方を担う組織として、歴史的に開発組織の「下流」と見なされてきました。
この「開発と運用」という切り口は相対的なものであり、実際には、「開発に踏み込んだ運用」を得意とする運用組織や、「自前の運用ツールの開発」を得意とする運用組織も存在してきました。
しかし、「コスト削減」や「専門領域の明確化」を名目に、運用組織の開発スキル部分を外部に切り出そうとする動きが後を絶ちません。
これは、その運用組織の特徴を失なうことを意味し、自ら「開発の下流工程化」を選ぶことに他なりません。
一方で、近年は1つのチームで開発と運用を行う「2ピザチーム」を選択する組織も増え、現行の運用組織にも開発スキルが期待される流れも強くなってきています。
本セッションでは、開発・運用分離の弊害、運用組織における開発視点の必要性について、解説および議論をしていきます

アジェンダ

序. 運用は下流?

1. 運用組織のパターン

2. 開発・運用分離の弊害

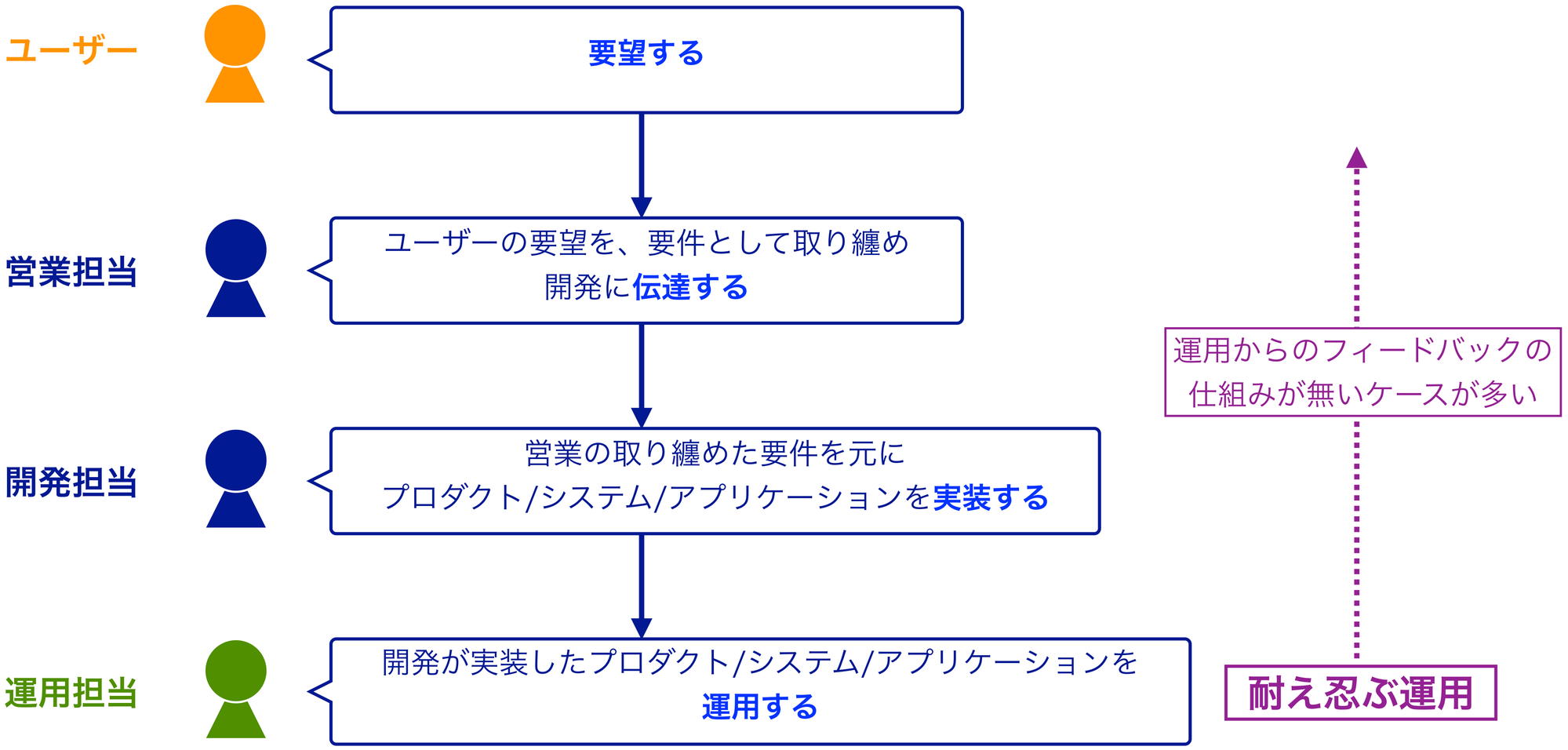
3. 運用組織観点の「開発」とは何か

4. 運用組織における開発視点の必要性

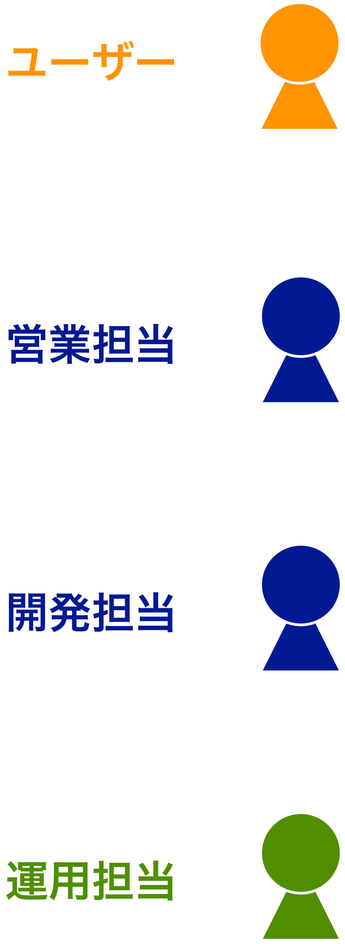
5. 今後の運用

序. 運用は下流?

運用は下流?


ユーザー
要望する
営業担当
ユーザーの要望を、要件として取り纏め
開発に伝達する
開発担当
営業の取り纏めた要件を元に
プロダクト/システム/アプリケーションを実装する
運用担当
開発が実装したプロダクト/システム/アプリケーションを
運用する
運用からのフィードバックの
仕組みが無いケースが多い
耐え忍ぶ運用

運用は下流?



本当に

運用は下流

なのか?

運用は下流?

ユーザー 

営業担当 

開発担当 

運用担当 

見方を考え直して
みましょう

「ビジネスモデル」の4要素

ビジネスモデル

利益を生み出す仕組み。

特に，情報技術やインターネットを利用して，消費者や取引先とのアクセス手段・商品や行為の選択・決済・配送まで一連の経済行為をシステム化し，さらにそれをモデル化したものを指す場合が多い。

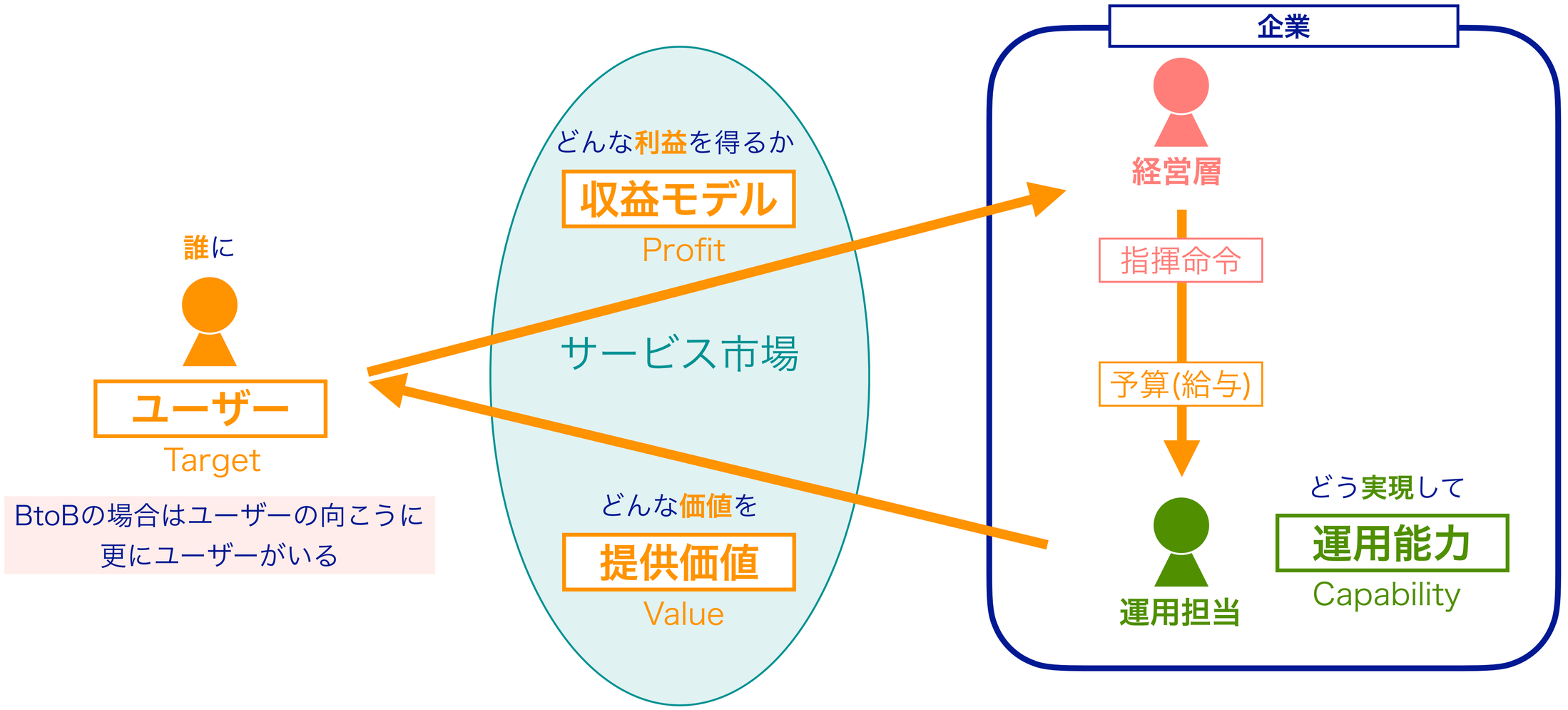
(出典: スーパー大辞林)

ビジネスモデルの4要素

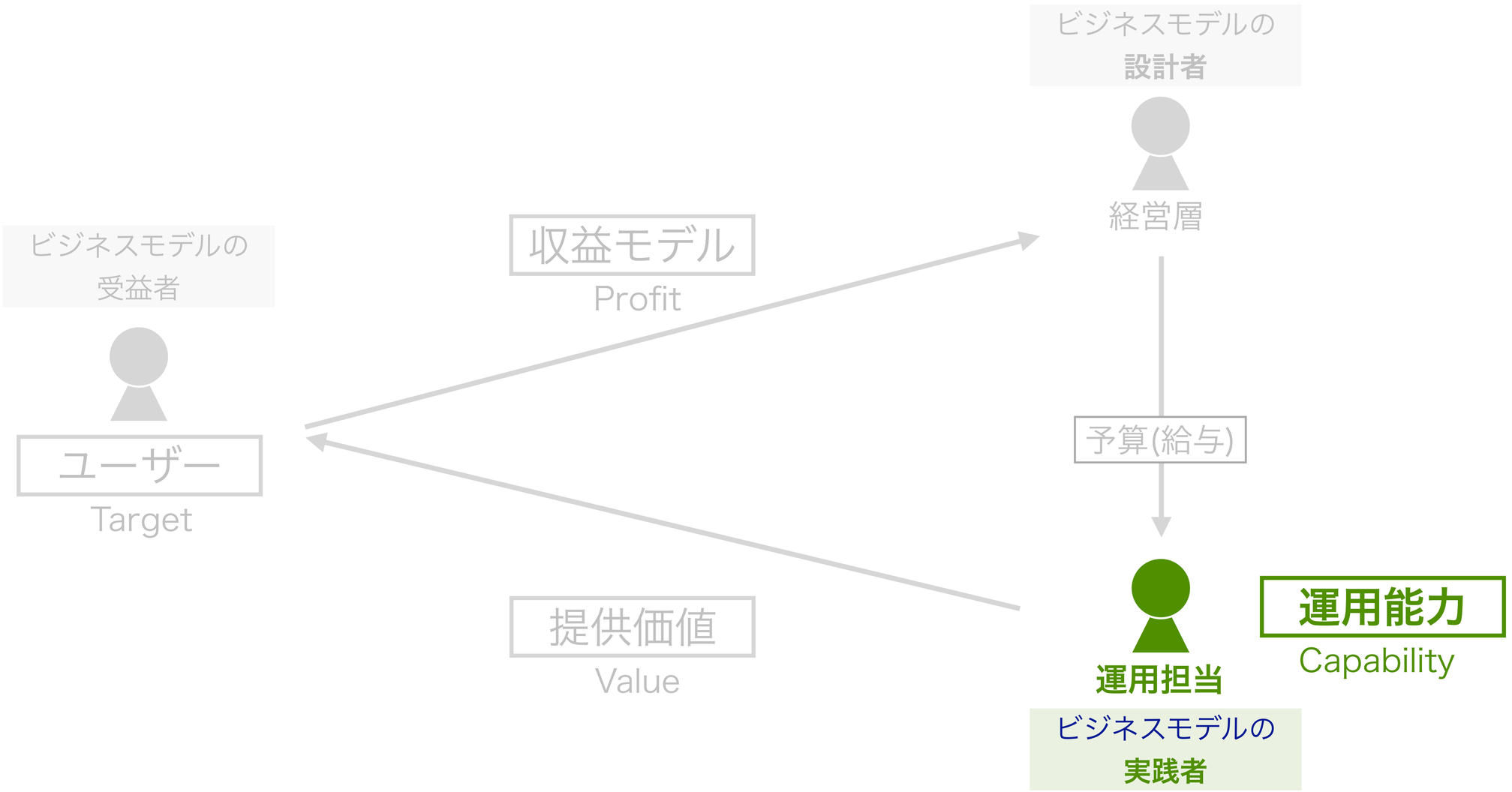
ユーザー Target	**誰**に	**提供価値** Value	どんな**価値**を
運用能力 Capability	どう**実現**して	**収益モデル** Profit	どんな**利益**を得るか

「ビジネスモデル」の基本構造

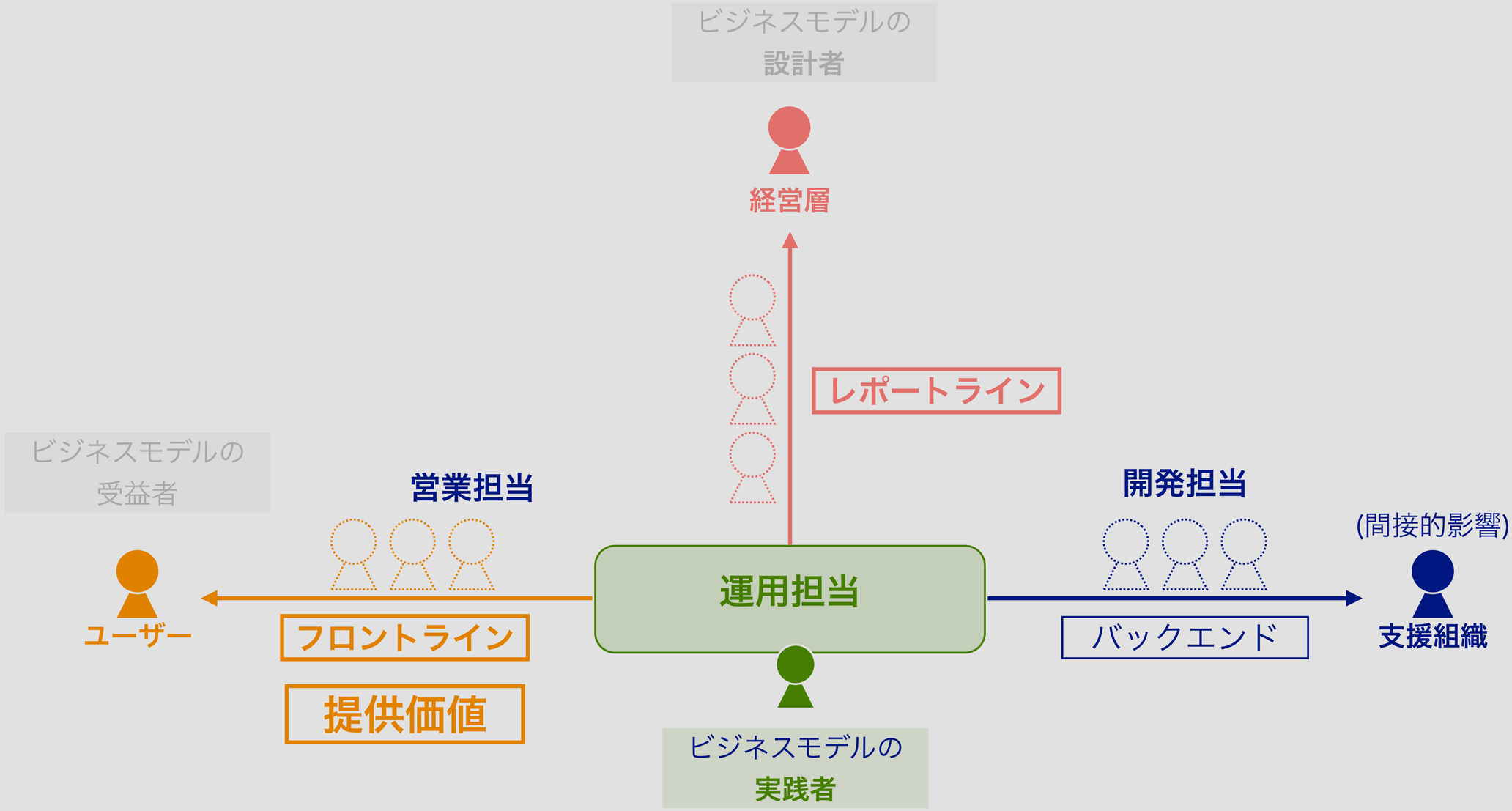
どんな**ユーザー**に、どんな**価値**を、どう**実現**して、どんな**利益**を得るか



ビジネスモデルにおける「運用」はビジネスのエンジン



参考: 「運用担当」から見る3つのライン (ステークホルダー)

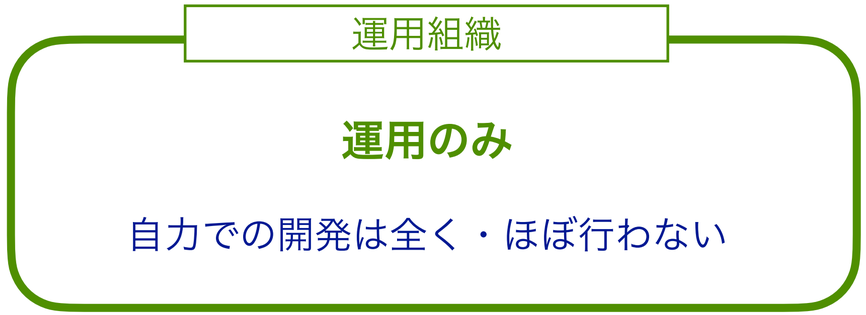


1. 運用組織のパターン

運用組織のパターン

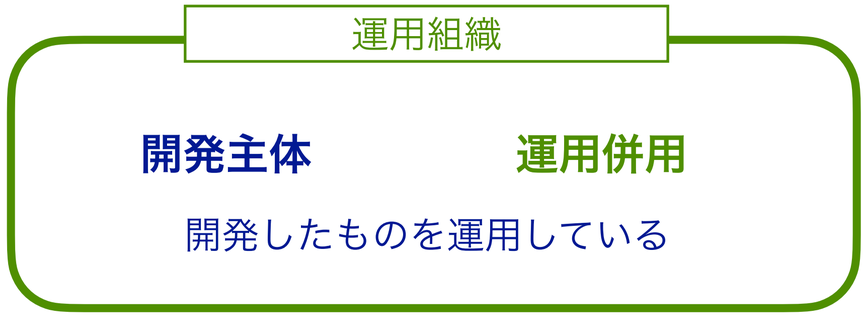
運用専任

運用専任型

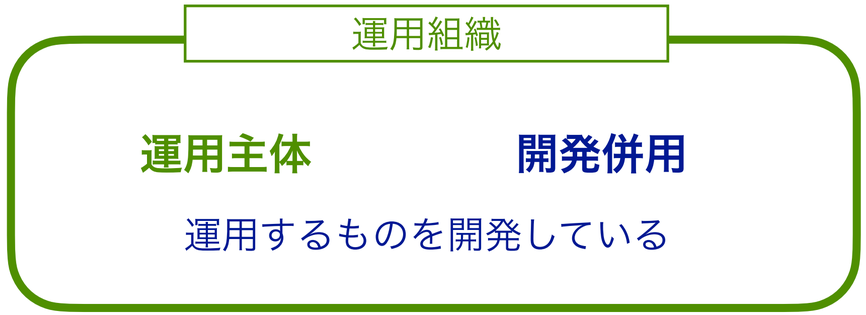


運用/開発兼任

開発主体型



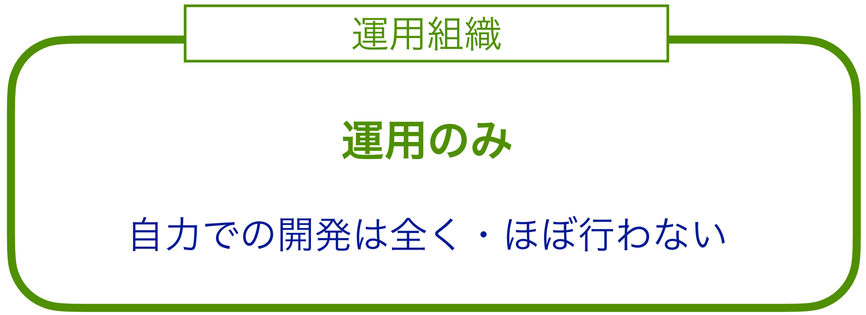
運用主体型



運用組織のパターン (運用専任型)

運用専任

運用専任型



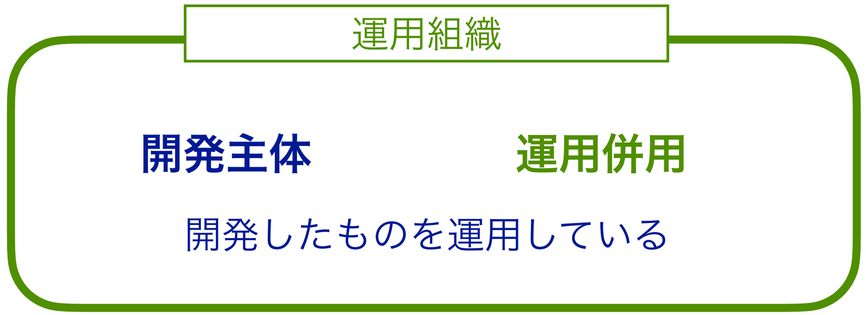
大企業に多いパターン。
本部レベルで専任になっていることも多い。

運用に明確な強みが無い場合、**コモディティ化し、下流になりやすい。**

運用組織のパターン (運用/開発兼任 開発主体型)

運用/開発兼任

開発主体型



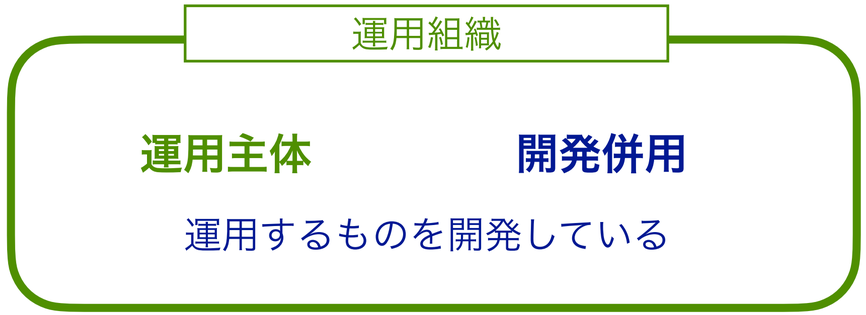
ベンチャー企業に多いパターン。
運用が業務範囲と明示されていない場合も多い。

実装がメイン業務で、運用はサブ業務のため、
リリースに追われて、**運用に手が回らない**ことも多い。

運用組織のパターン (運用/開発兼任 運用主体型)

運用/開発兼任

運用主体型



委託運用や**MSP事業者**に多いパターン。
開発が業務範囲と明示されていない場合も多い。

自力で大規模な実装はできないが
ユーザーのプロダクトの改修や**内製ツールの実装**はできることが多い。

運用の理想的な形態だが、人材の育成が難しい

2. 開発・運用分離の弊害

専門性が混在しているように見えるため開発と運用を分離しがち

特に、大企業では
機能による分割の誘惑に負けやすい。

どこでもある開発組織と
どこにでもある運用組織が
誕生する結果となりやすい

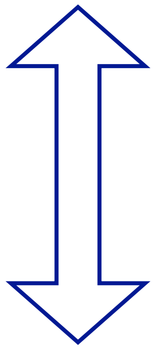
運用と開発を**一気通貫に見ることができる**
強みが失われる

組織は分けると乖離していく

開発担当

開発のことしか考えない。運用の苦労を知らない。

自組織第一主義に陥り、
立場や考え方が乖離していく。



関係が疎遠になり、
コミュニケーションコストが増大する。

運用担当

運用のことしか考えない。開発の苦労を知らない。

開発・運用分離の弊害

開発担当

運用手法や**運用の苦労**を知らない。

一方的な依頼になりがち

運用現場の現実に合致せず、
効果的な運用につながらない。

言われた仕事が
中心になりがち

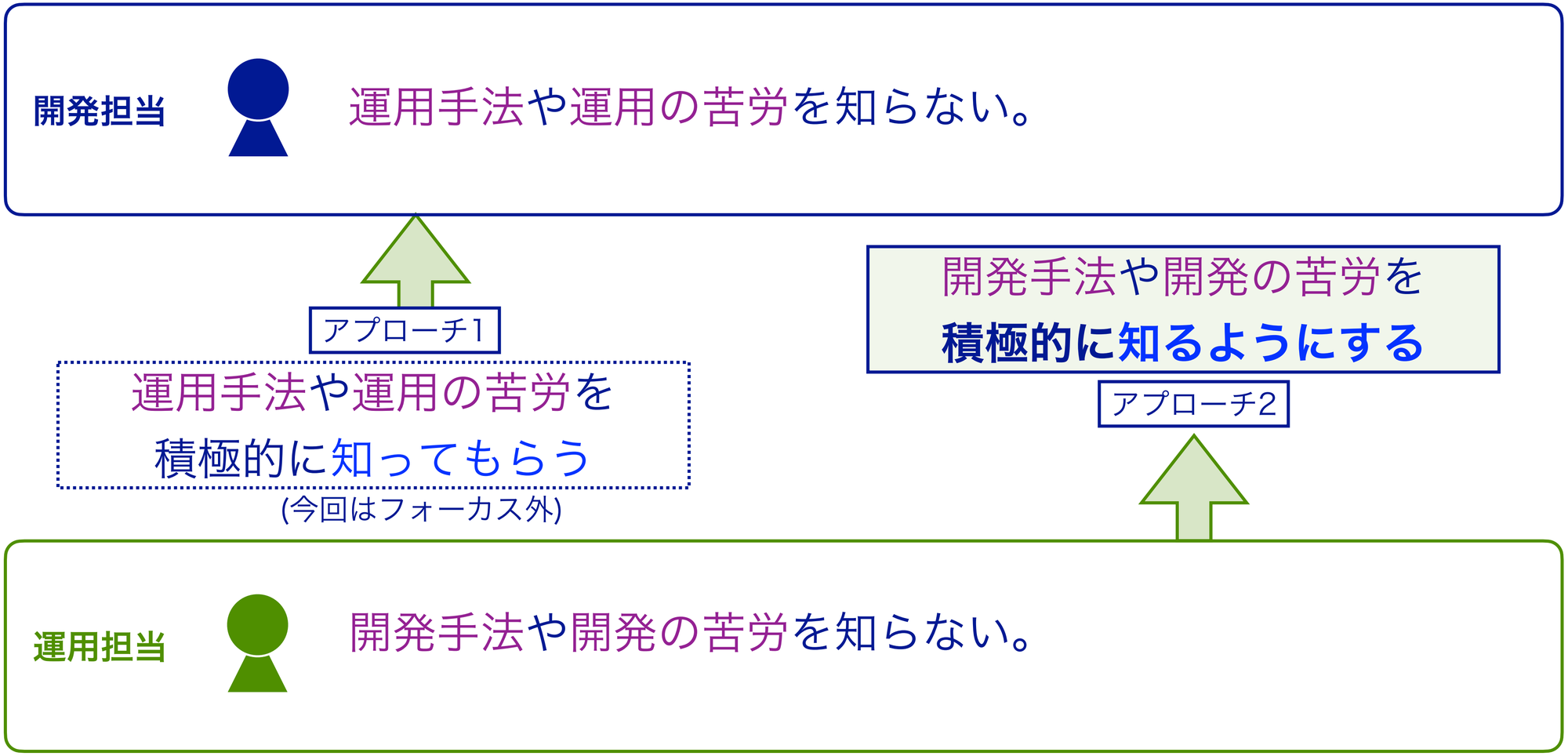
フィードバックが効かない

運用担当

開発手法や**開発の苦労**を知らない。

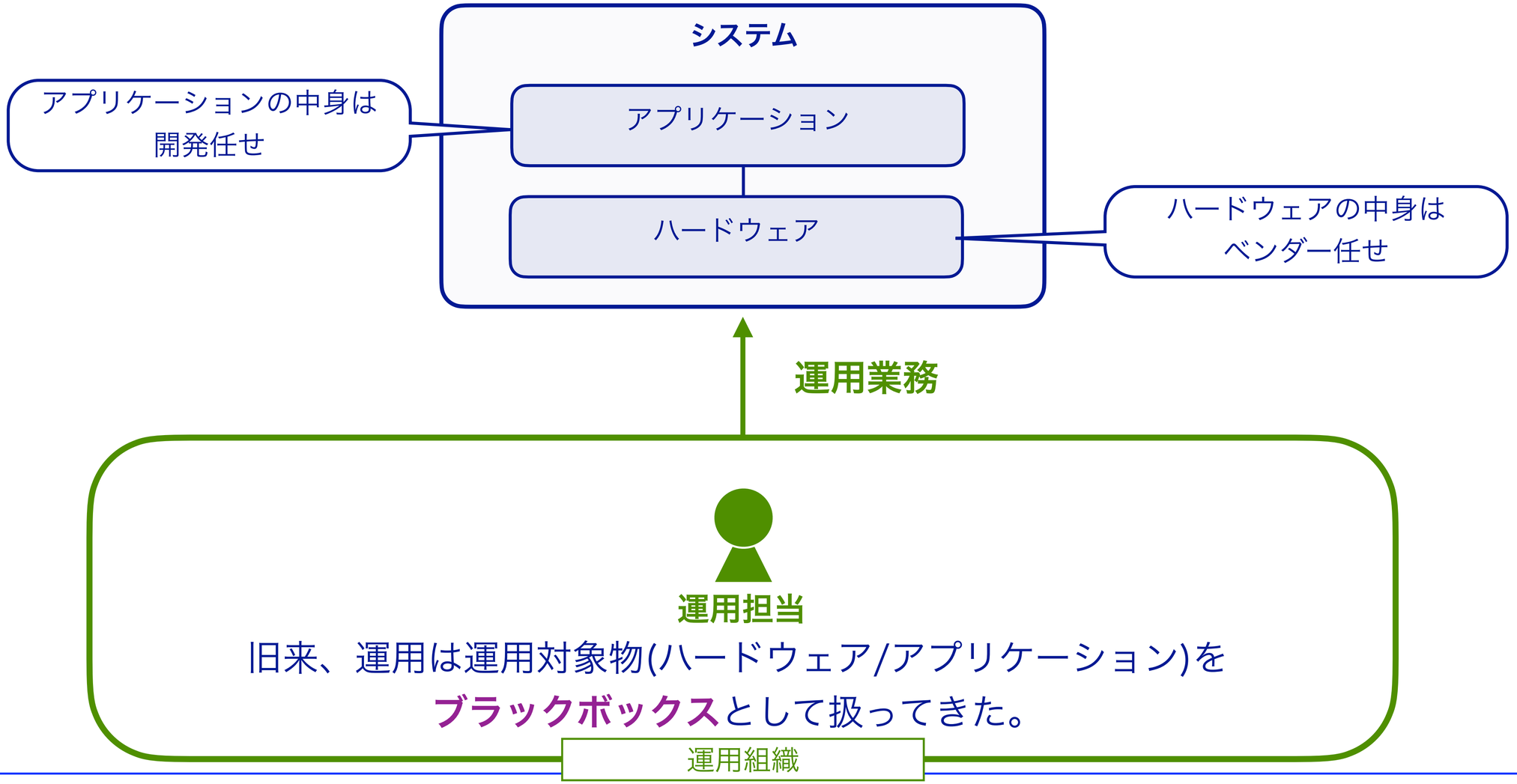
自分の道具を自分で作れない。作れてもやり方や考え方が古い。維持や更新ができない。

弊害を解消するための2つのアプローチ



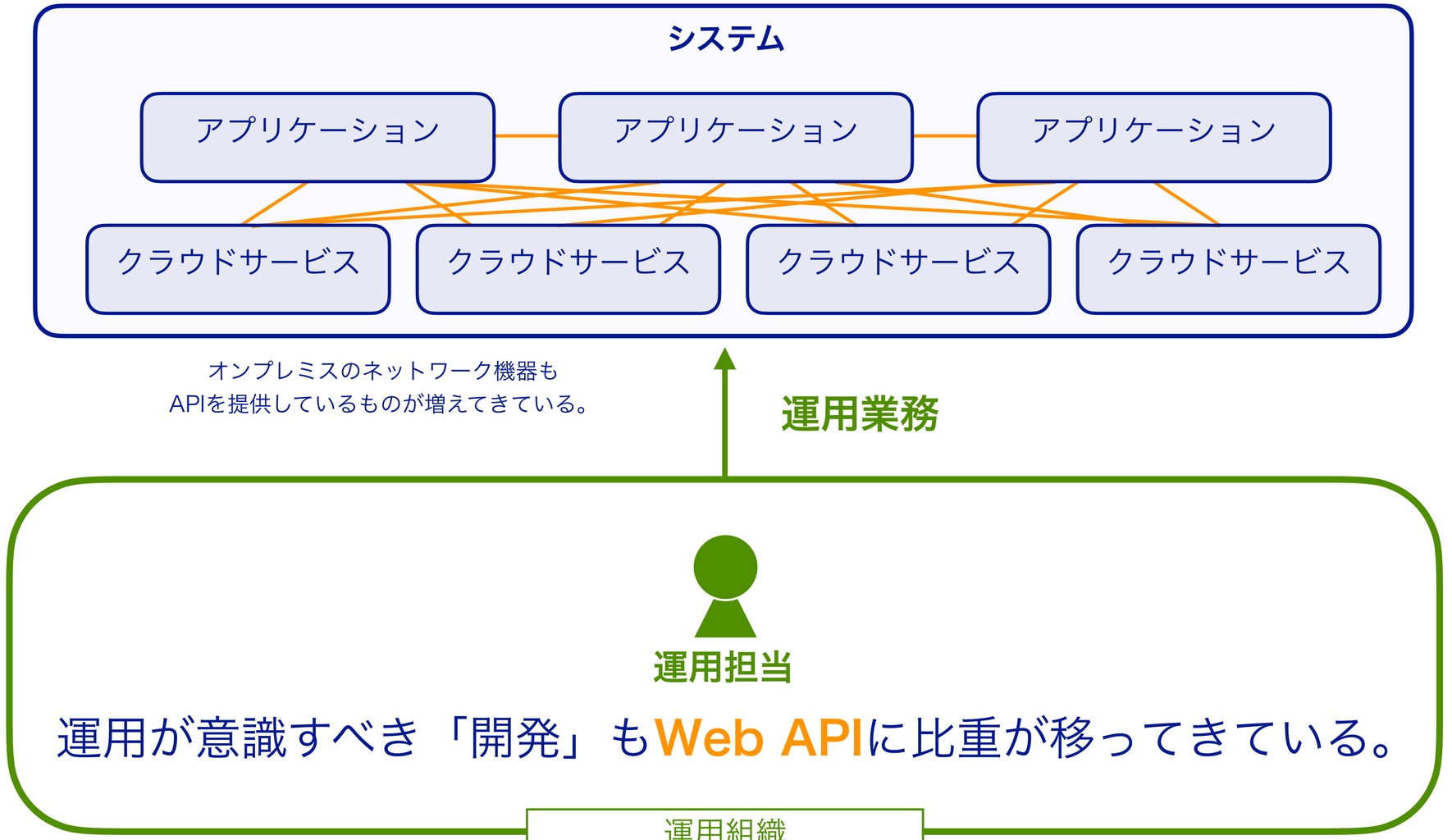
3. 運用組織観点の「開発」とは何か

運用組織から見る「開発」の変化 (オンプレミス)



運用組織から見る「開発」の変化 (クラウド)

クラウド運用の比重が高くなり、**Web APIによる連携**が**一般化**している。

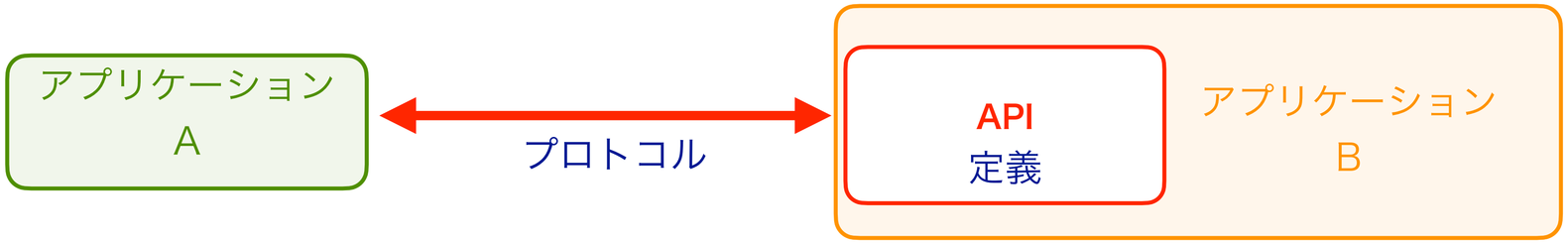


APIとは

API　**Application Programming Interface**

アプリケーション・ソフトウェアを
構築し、統合するための**一連の定義とプロトコル**であり、
ソフトウェア同士を**つなぐ(連携させる)こと**で、
アプリケーションの開発を容易にします。

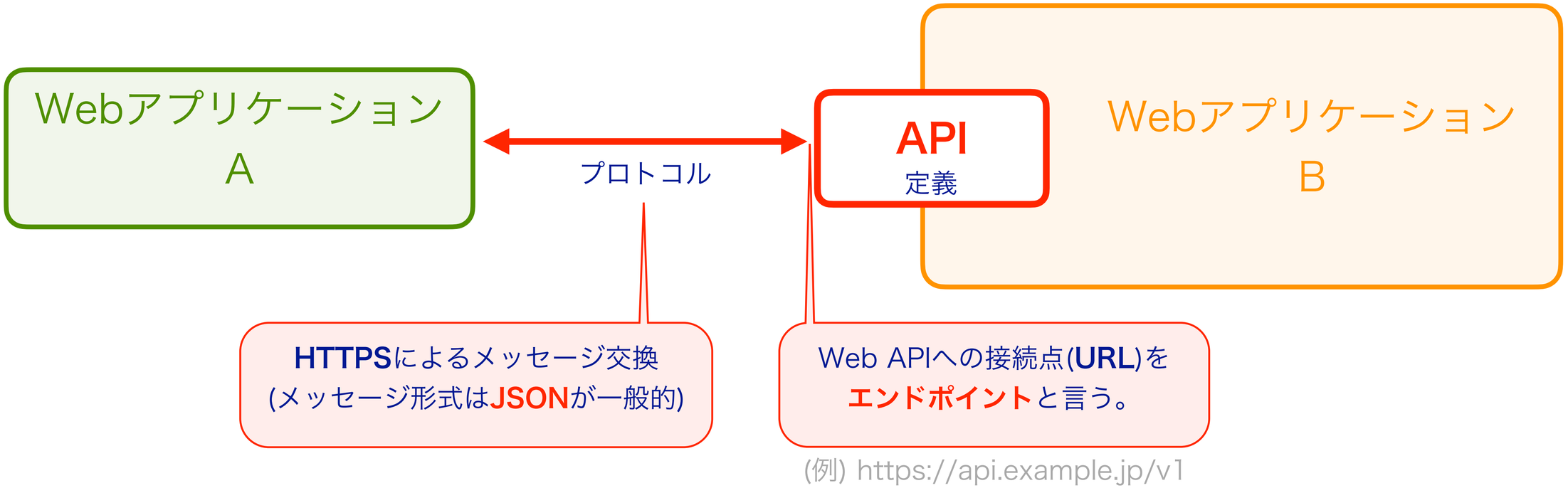
(出典: RedHat APIとは https://www.redhat.com/ja/topics/api/what-are-application-programming-interfaces)



交換されるメッセージは、**メモリ上のポインタやバイナリ**であることが一般的

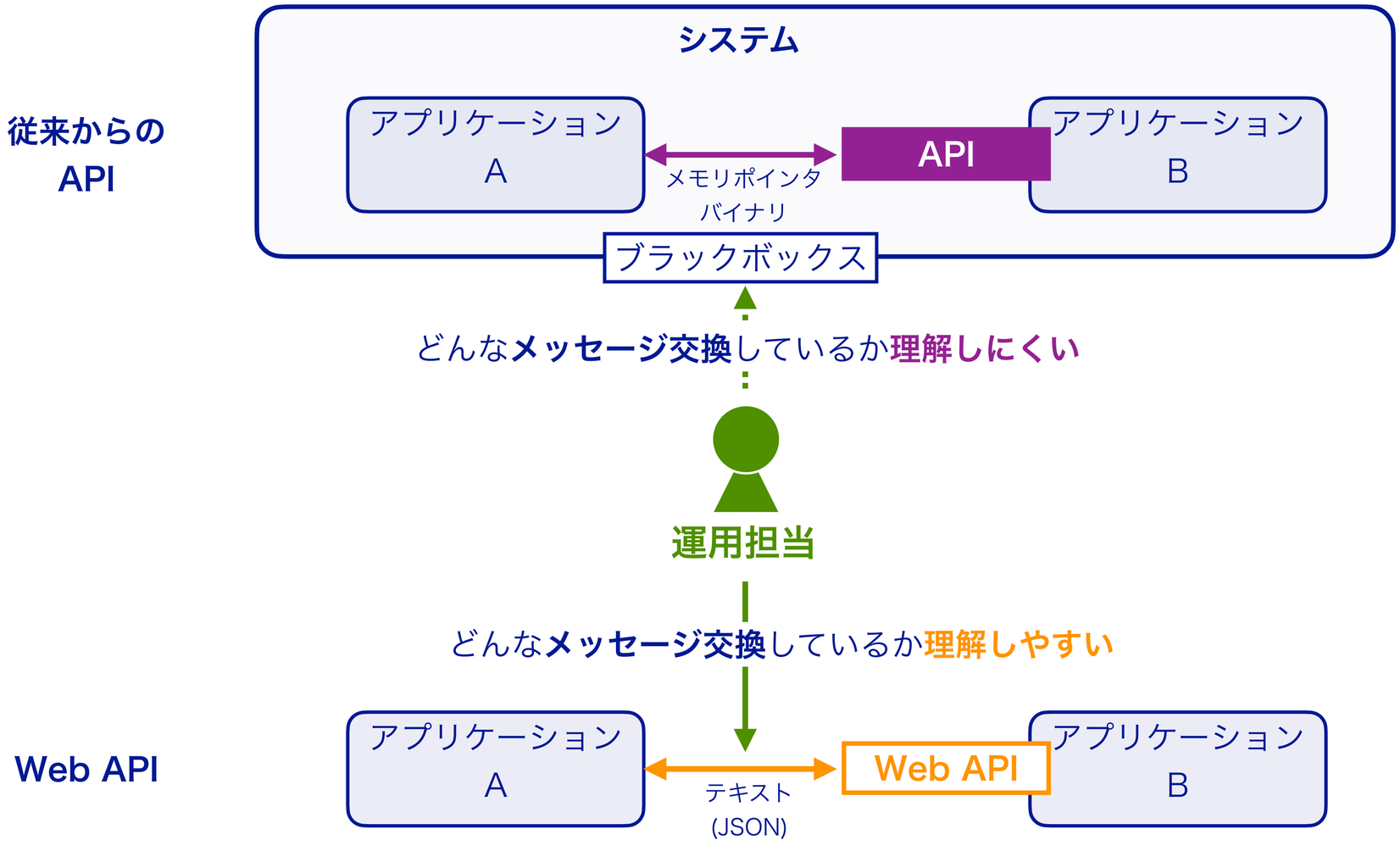
Web APIとは

Web技術を活用したAPI



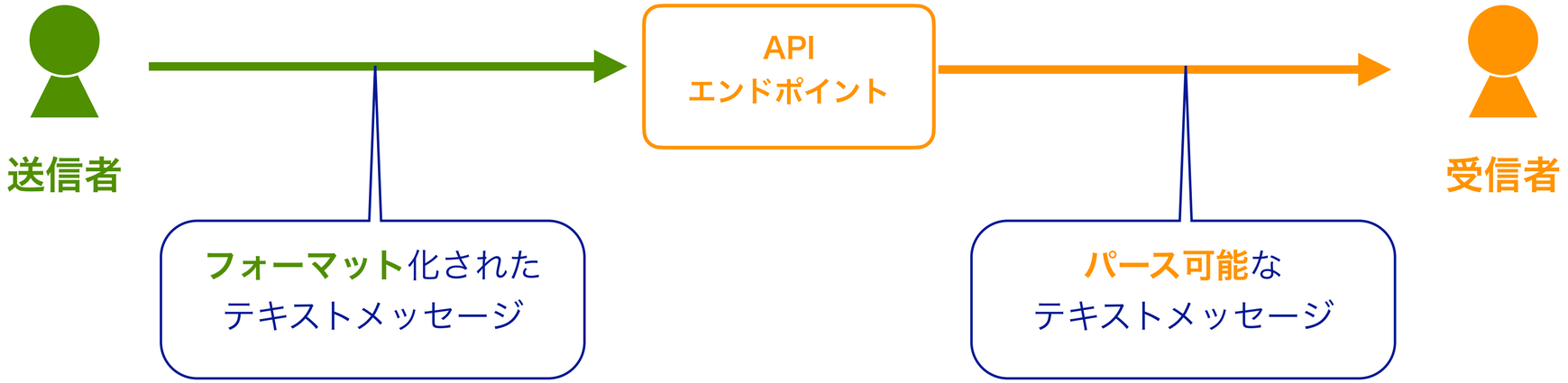
交換されるメッセージが、**可読性のあるテキスト**であることが特徴

従来からの「API」と「Web API」の違い



Web APIの基本モデル

オブジェクト同士を**テキストデータでつなぐこと**で、
その**連携を容易**にする。

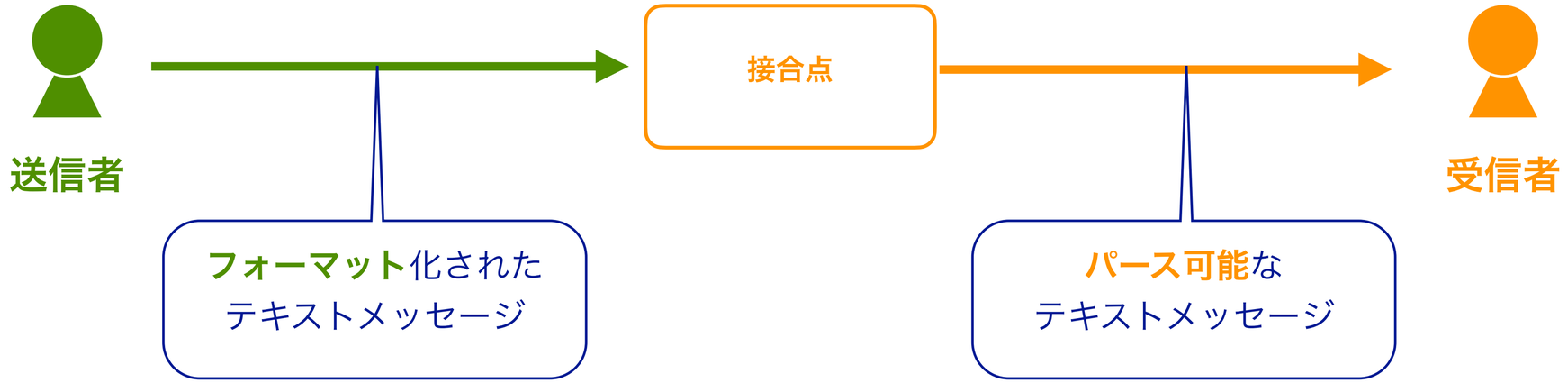


送信側は
決められたフォーマットで送信するだけ
で良い

受信側は
パース可能なフォーマットで受信するだけ
で良い

Web APIは「全く新しいアイデア」ではない

オブジェクト同士を**テキストデータでつなぐこと**で、
その**連携を容易**にする。

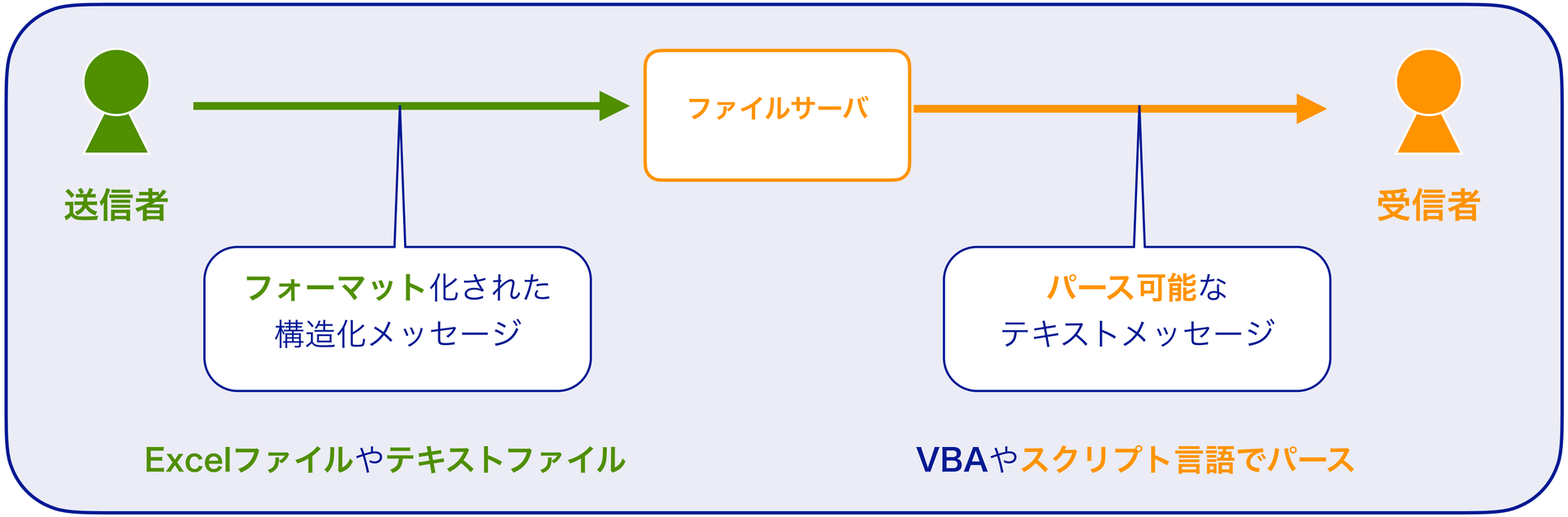


大事なのは、**メッセージ交換の接合点の指定**と、**交換するメッセージの定義**

既存のアイデア例1. パース可能なファイルによる連携

フォーマット化されたファイルを

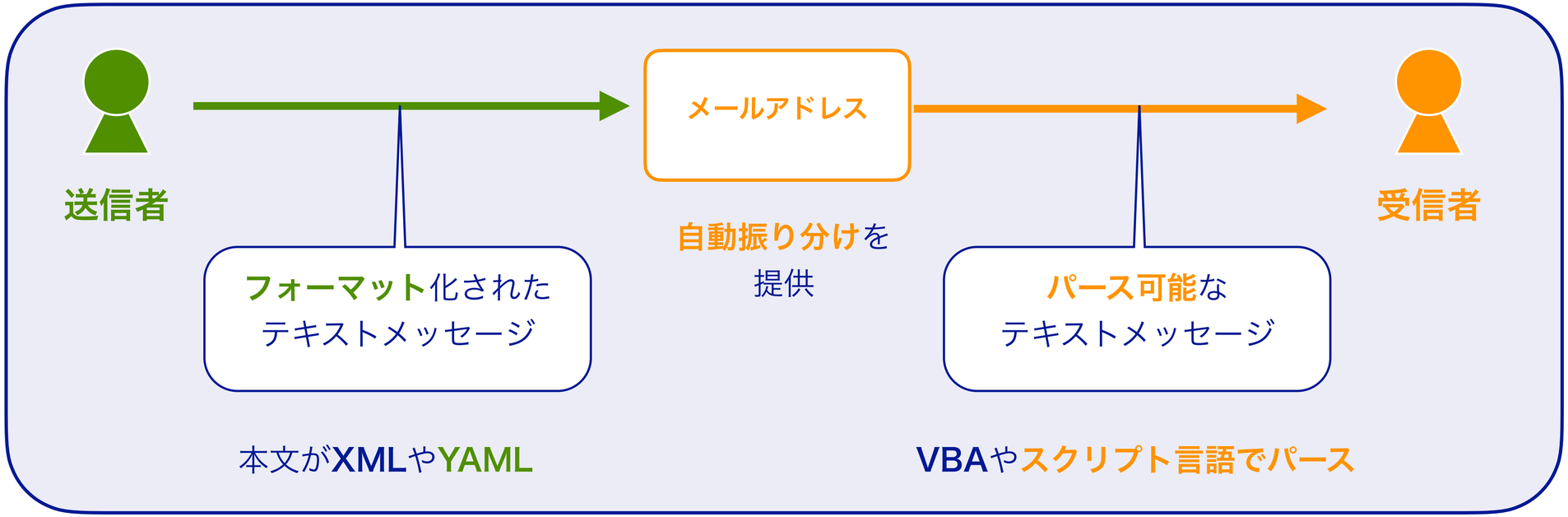
ファイルサーバというエンドポイントに送信する。



既存のアイデア例2. パース可能なメールによる連携

本文がフォーマット化されたメールを

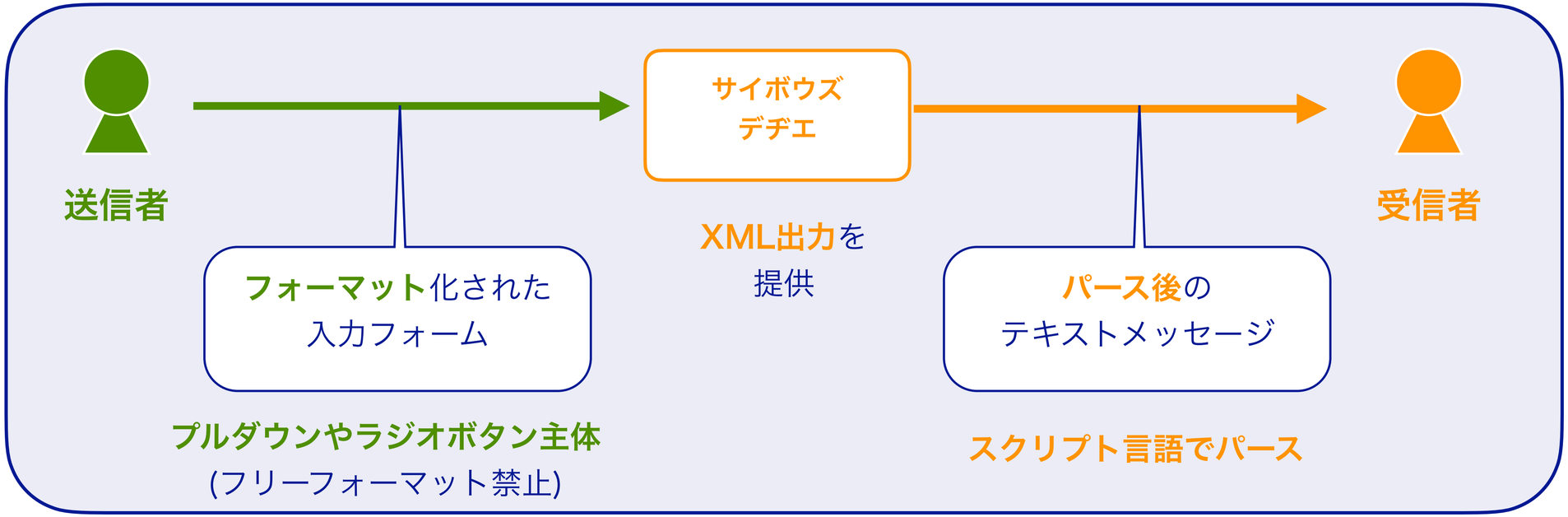
メールアドレスというエンドポイントに送信する。



既存のアイデア例3. パース可能なWeb UIによる連携

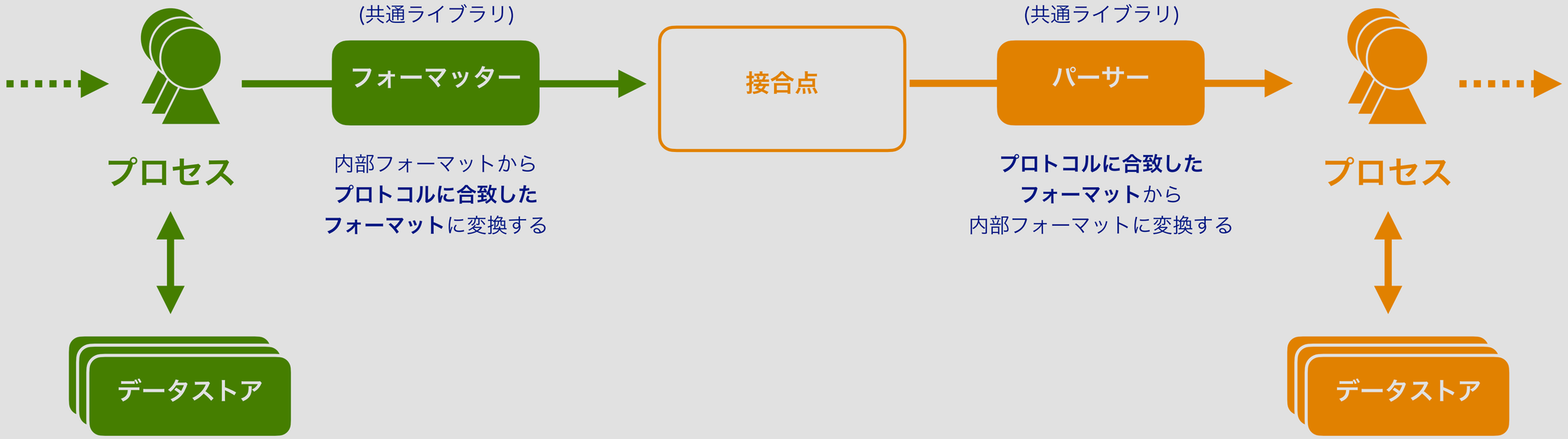
入力データがXMLで出力されるWebツールで

後続のプロセスに**パース可能な状態で渡す**



参考: テキストメッセージ交換の基本モジュール

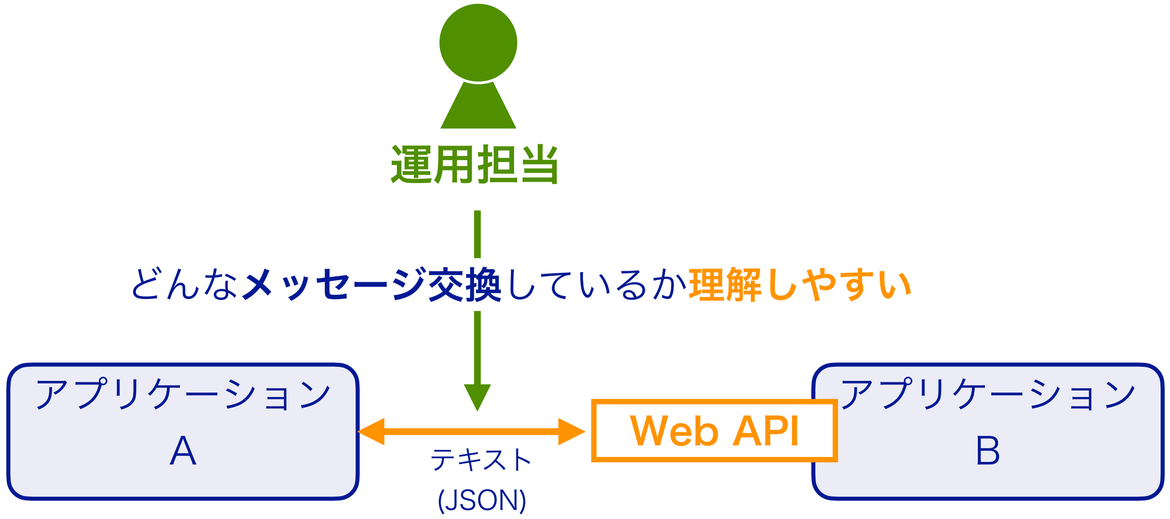
接合点を介して、**出力側にフォーマッター**、**入力側にパーサー**を配置する



同一プロトコルであれば、共通ライブラリとして反復利用できる。

Web APIの本質はテキストメッセージの交換

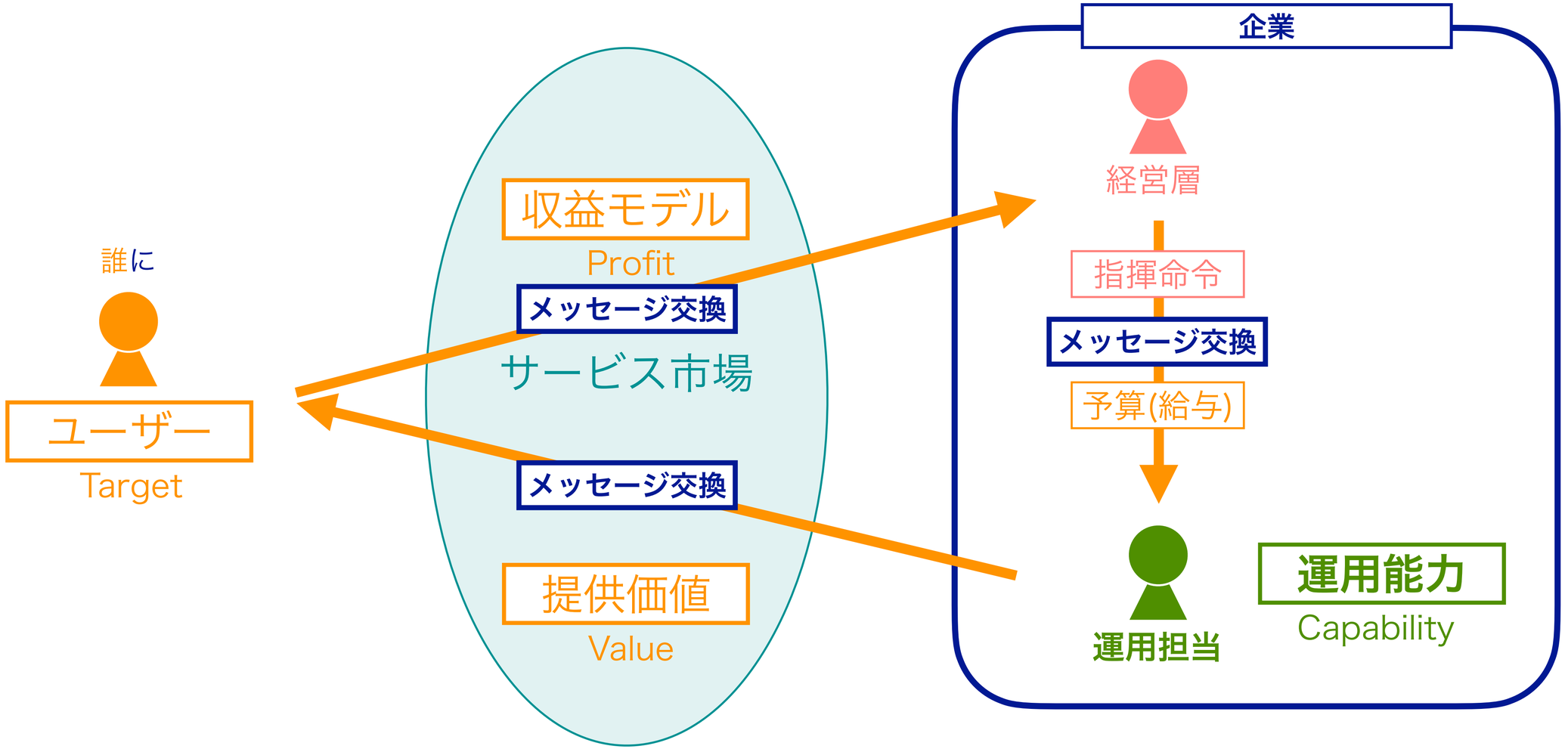
システムが、どのようなテキストで**メッセージ交換**をし、
全体として、どのような**データフローが行われているか**を知ることが大事



Web API　メッセージ交換の接合点の指定 + 交換するメッセージの定義

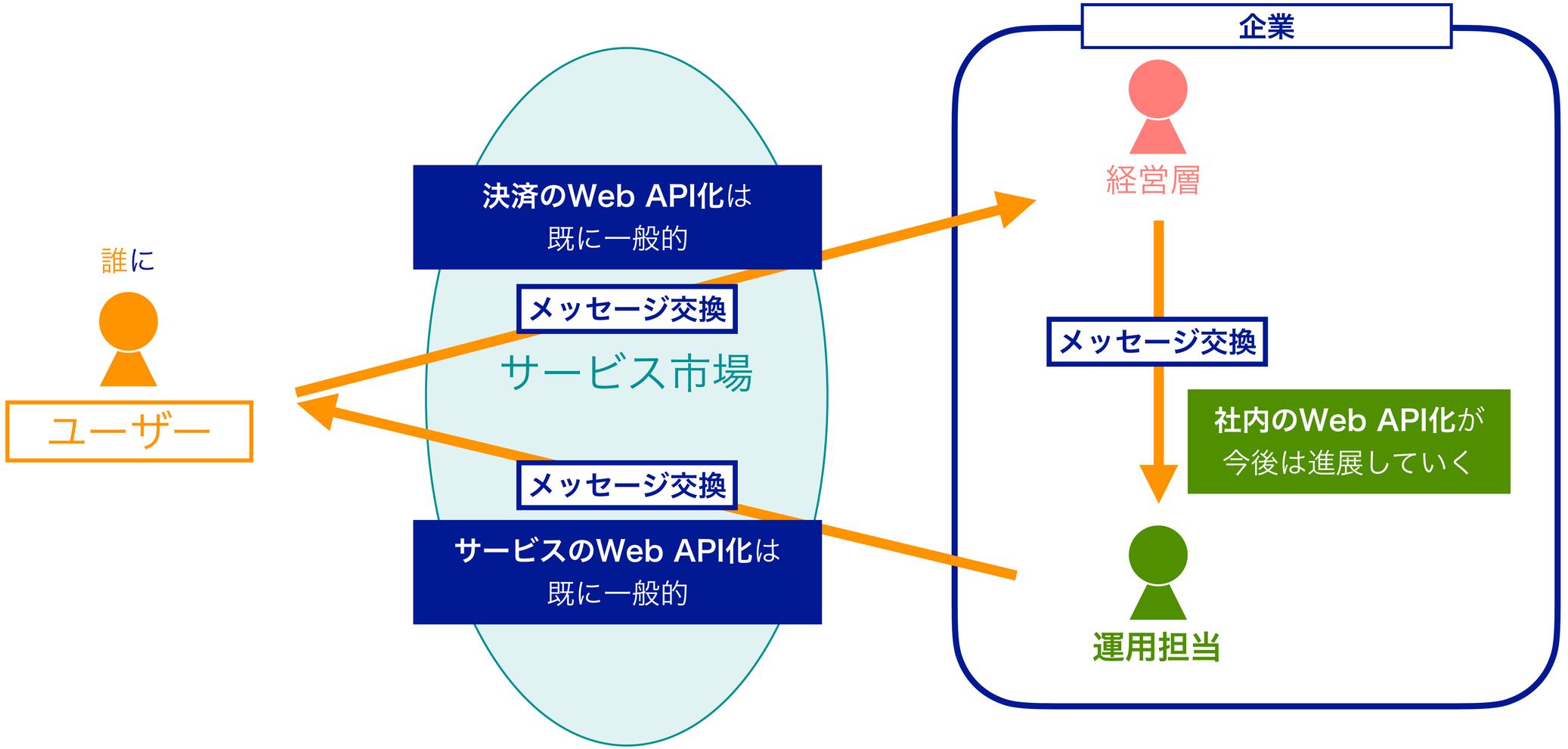
ビジネスモデルもメッセージの交換

どんなユーザーに、どんな価値を、どう実現して、どんな利益を得るか

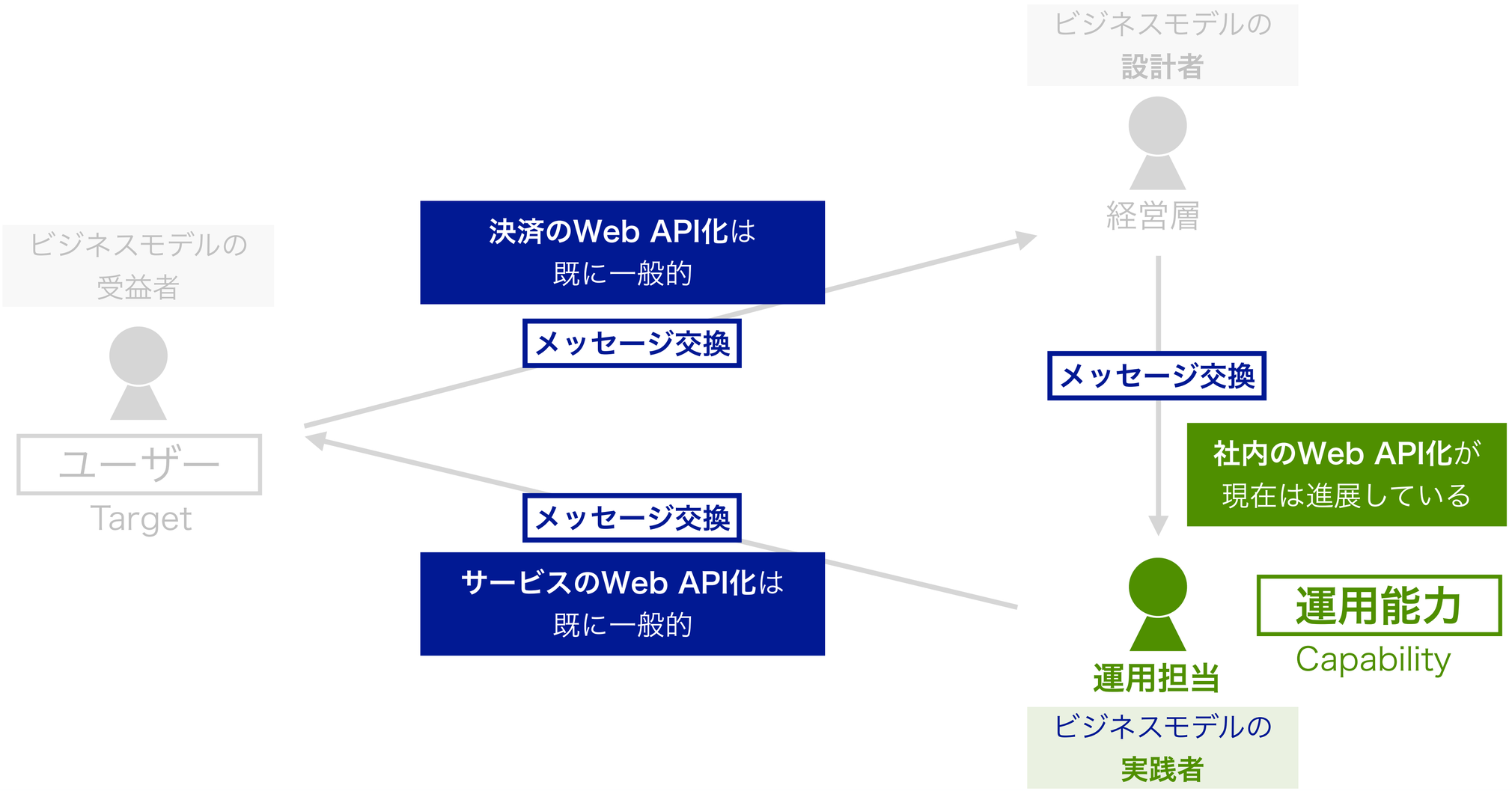


ビジネスモデルにおけるWeb APIの普及

サービス市場におけるWeb APIの普及は定着。今後は企業内に進展。



運用組織もWeb API技術に慣れ親しんでいく必要がある

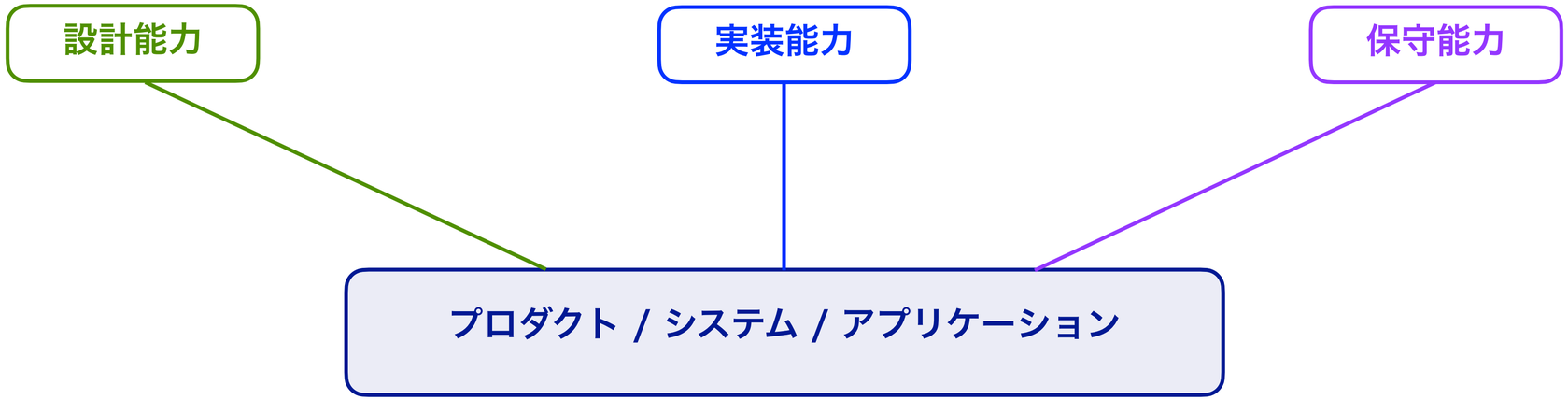


運用組織観点の「開発」とは何か (従来)

運用にとっての「開発」

運用に関わるプロダクト/システム/アプリケーションを、**設計・実装・保守**すること

および、それらを**遂行する能力**



内製ツール以外は、**運用にとってブラックボックス**であることが多かった

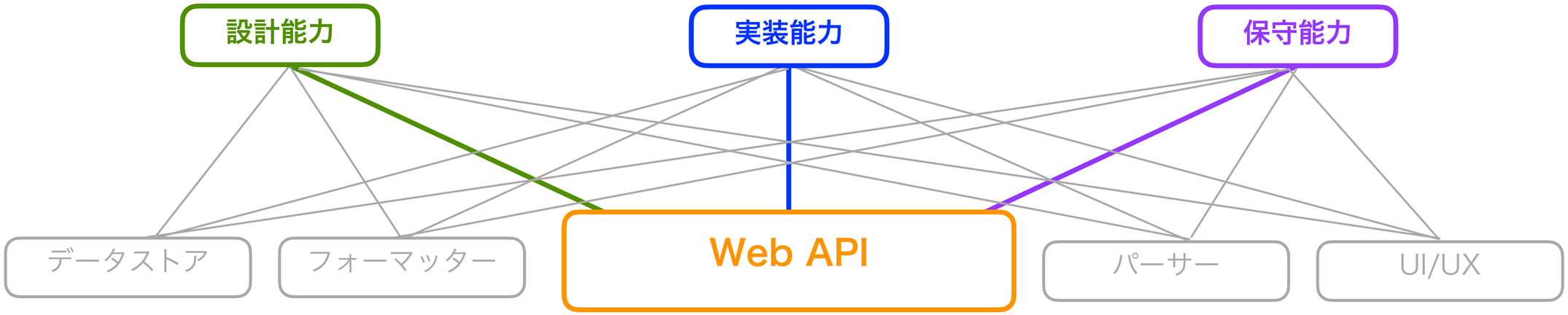
- 運用対象のシステム、アプリケーション
- 顧客から運用委託されているプロダクト
- 運用組織の基盤システム

運用組織観点の「開発」とは何か (今後)

運用にとっての「開発」

運用に関わるプロダクト/システム/アプリケーションを、設計・実装・保守すること

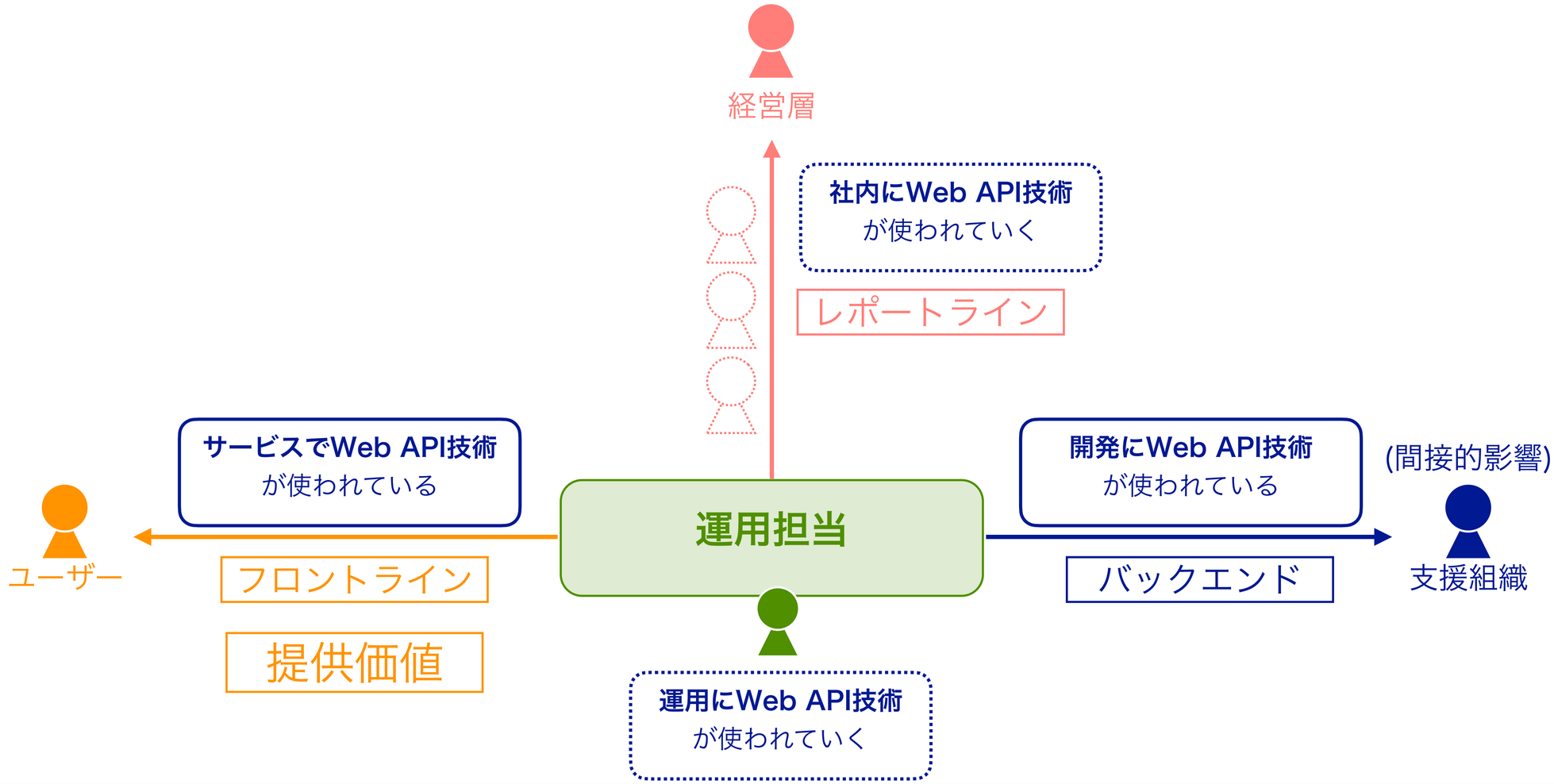
および、それらを遂行する能力



Web APIに関わる技術を活用していくこと

が、主眼となっていく　運用対象の**ホワイトボックス**化

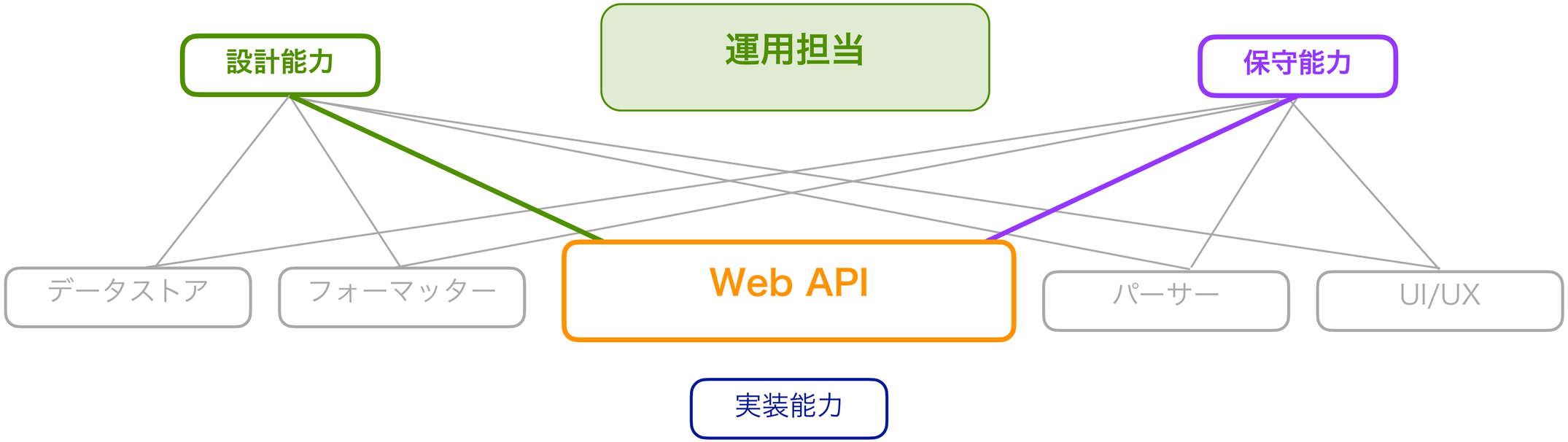
今後の運用は積極的にWeb API技術と関わっていく



4. 運用組織における開発視点の必要性

運用組織における開発視点の必要性

運用組織にも、Web APIの設計能力・保守能力が必要



自前で実装できなくても、**少なくとも設計**、可能なら保守できることが望ましい。

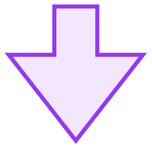
運用組織がWeb APIの設計能力を持つ必要性

設計能力

対内的

運用を取り巻く環境

常に変化し続ける。



変化への対応を迅速に求められる。

能動的に運用を行っていくためには、**内製が不可欠**となる。

汎用部分は**製品活用**

コアコンピタンス部分は**内製**

運用担当

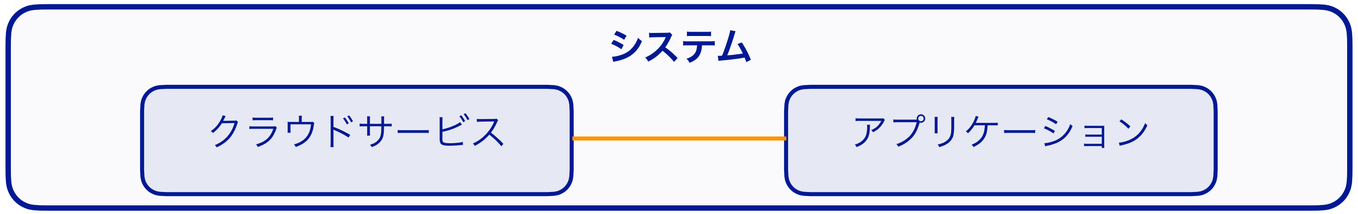
プロフェッショナル組織であれば、**自分達の道具は、自分達で作る。**

実装は支援組織に委託するとしても、必ず**設計は自前**でする。

運用組織がWeb APIの設計能力を持つ必要性

設計能力

対外的



運用対象プロダクトを
技術的に深く知ることができる。
(開発担当との共通認識)

運用対象の今後(ビジネスや技術の動向)
を予測しやすくなる。
(経営層や開発担当との共通言語)

運用組織の設計能力は、**対外的に大きな副次効果**をもたらす。

運用担当

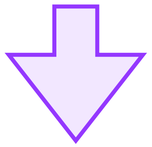
プロフェッショナル組織であれば、**自分達の運用対象を熟知する。**

自分達の実装を外部委託している場合、委託側の観点でプロダクトを見ることもできるようになる。

運用組織がWeb APIの保守能力を持つ必要性

保守能力

運用を取り巻く環境　常に変化し続ける。



変化への対応を迅速に求められる。

内製を適切に維持・拡張するためには、**保守能力が不可欠**となる。

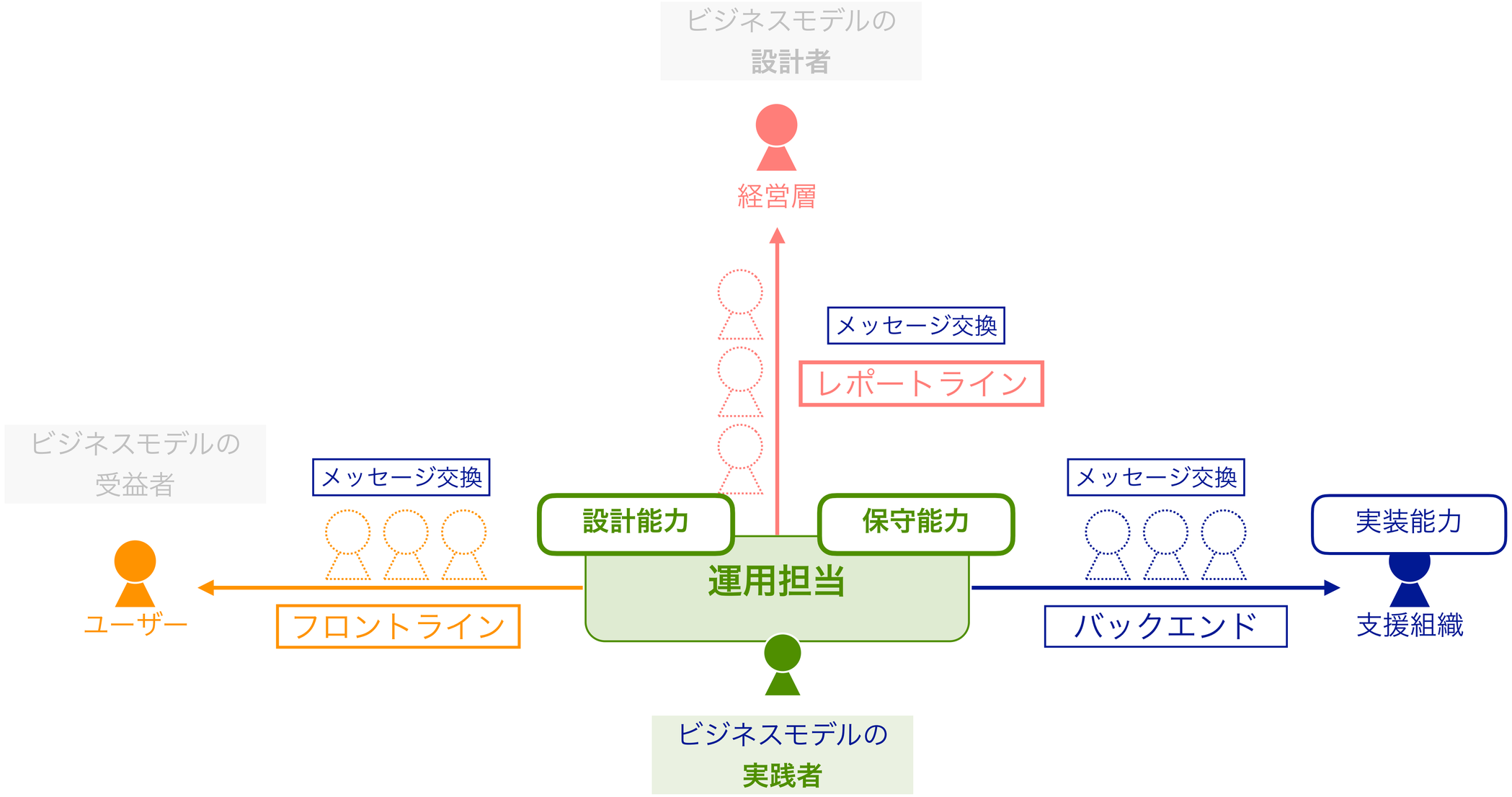
新たな需要への対応	現状とのギャップの解消
攻めの保守	守りの保守

運用担当

プロフェッショナル組織であれば、**自分達の道具は、自分達で保守する。**

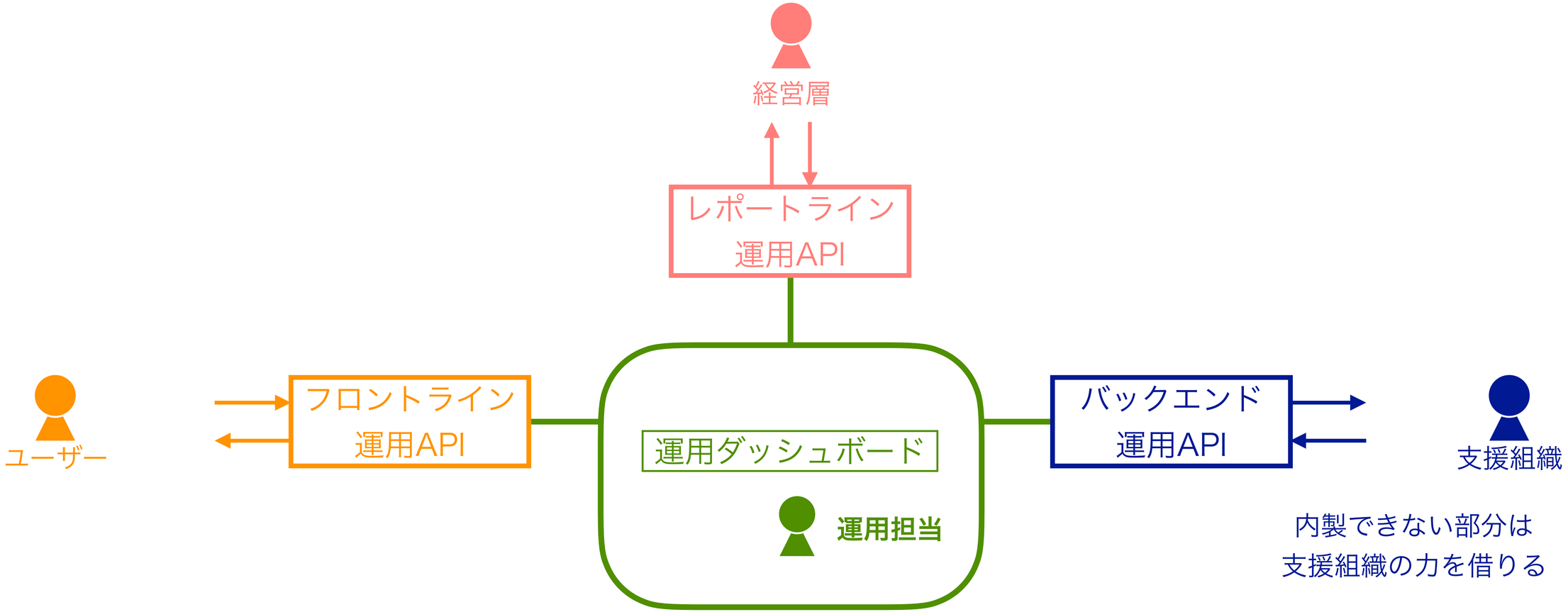
実装の修正は支援組織に委託するとしても、必ず**設計の変更は自前**でする。

運用3ライン図におけるWeb APIの設計・実装・保守



運用基盤のコア部分を内製化

外部に対して「**運用API**」で**情報を収集・提供**し、
ダッシュボードなど直接触るものは**自力で加工しよう**



5. 今後の運用

今後の運用が持つべき開発能力

開発エンジニアとの共通言語を身に付けていく

設計能力

オブジェクト指向分析・設計、リリースマネジメント
Web API/データストア/フォーマッター/パーサーなどの**概要設計、詳細設計**
テスト設計

実装能力

オブジェクト指向プログラミング
Web API/データストア/フォーマッター/パーサーなどの**実装**
テスト実装

保守能力

設計能力と類似の能力
(ボトムアップ アプローチ)

共通

開発に必要な一般教養
情報科学、モデリング技法、開発技法、テスト技法
コーディング規約、分散バージョン管理フロー
プログラミング言語、ライブラリ、フレームワーク、など

開発との共通認識を育てていくためには

開発担当　　**運用手法**や**運用の苦労**を知らない。

開発と運用の
両方を経験している人が
最も運用に向いている

開発の流儀を知る。
流儀は変わりつづける。
変わる前提で学び続ける。

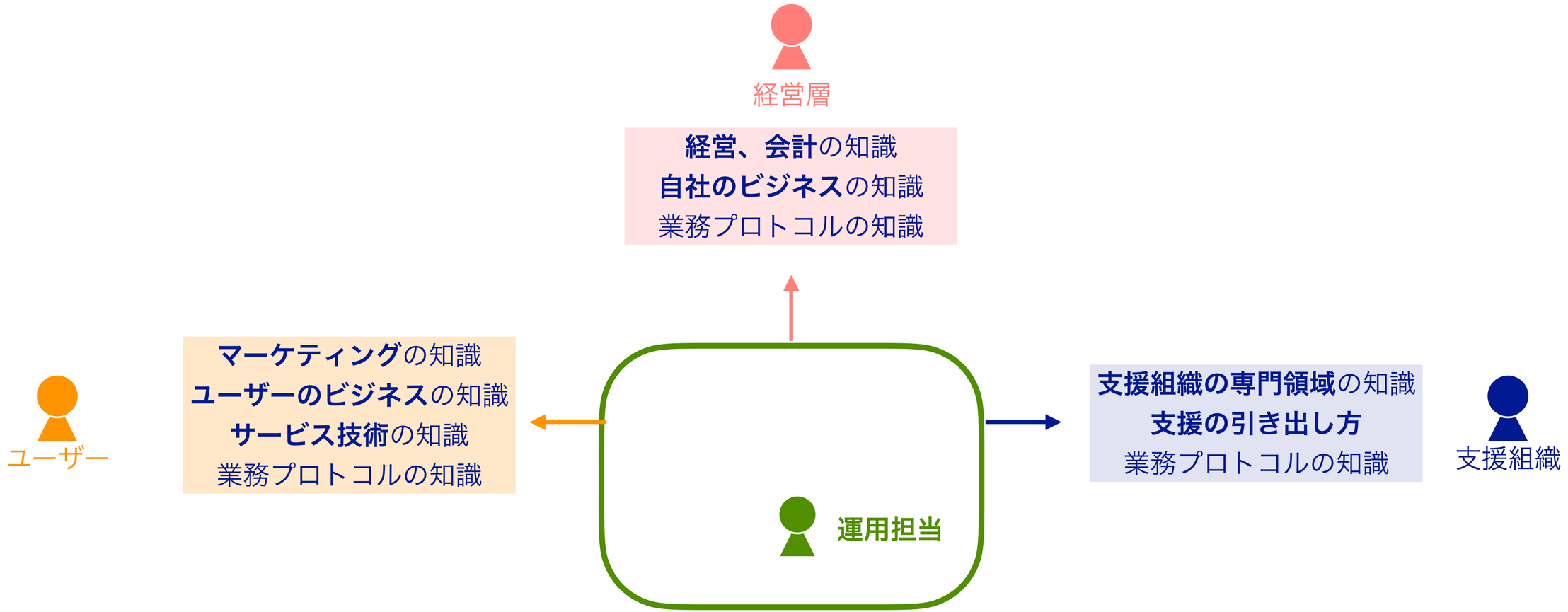
運用担当

開発手法や**開発の苦労**を知らない。

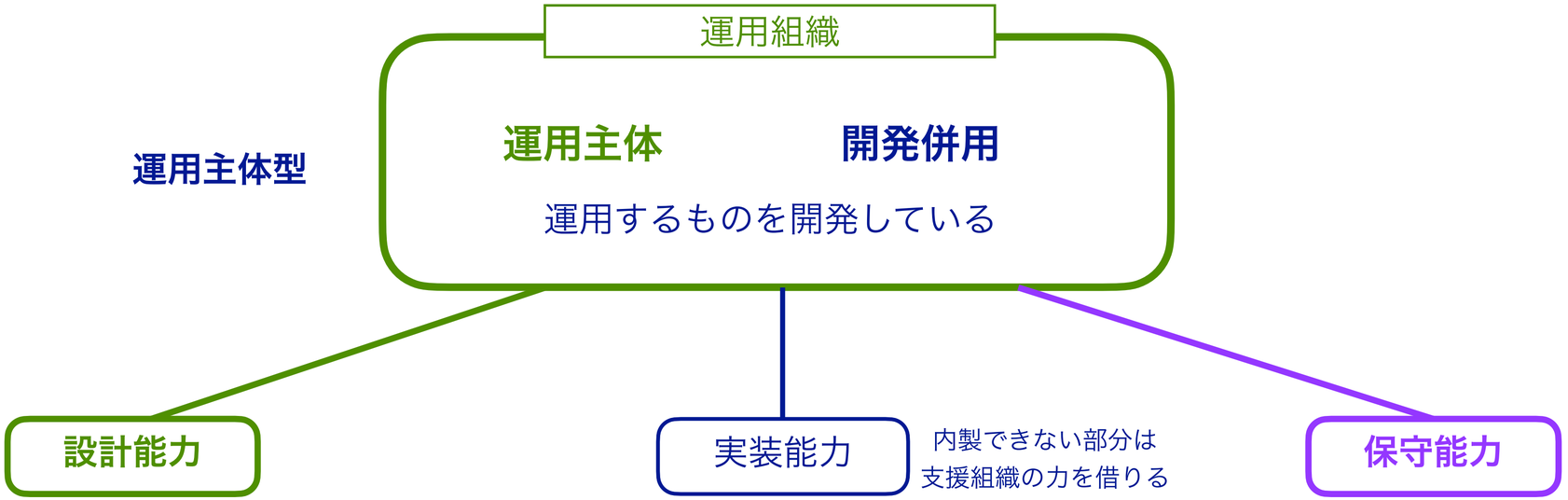
運用の3つのラインと共通言語を持つ

一緒に仕事するステークホルダーとの共通言語を身に付けていく

(開発の方向性が迷走しないために必要)



まとめ: 今後の運用



自分達の道具は、自分達で作り、自分達で保守する。

自分達の運用対象を熟知する。

＋

一緒に仕事するステークホルダーとの共通言語を身に付けていく

参考: 過去の発表資料

過去の発表資料は
OpsLab.jp というサイトに置いてあります。

https://www.opslab.jp/publish/

Operation運用設計Lab

http://www.operation-lab.co.jp/