

伊藤忠テクノソリューションズ株式会社様

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

運用設計導入コース (第1回)

運用フレームワークの考え方

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

運用設計ラーニング

2022-12-14

OpsLearn
運用設計ラーニング

アジェンダ

- ・ 導入1. 学習ガイド
- ・ 導入2. 講師の経歴・運用経験
- ・ 導入3. 運用あるある
- ・ 本編
- ・ まとめ

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
当日配布します
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn
運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

導入1

OpsLearn

OpsLearn

学習ガイド

運用設計ラーニング

運用設計ラーニング

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearnの目的

従来、現場ごとの個別事情によりやり方が異なるため、標準化が難しいと言われてきた「運用」

「運用現場視点による運用設計のための方法論」により
みなさんの運用現場における「実践的な運用設計」への取り組みを促進すること

▶ サービスの安定

社会基盤に相応しい安定運用。

「安定した運用」の実現

▶ 業務負荷の平準化

個々人ががんばりすぎなくてもうまく業務が回る運用現場。

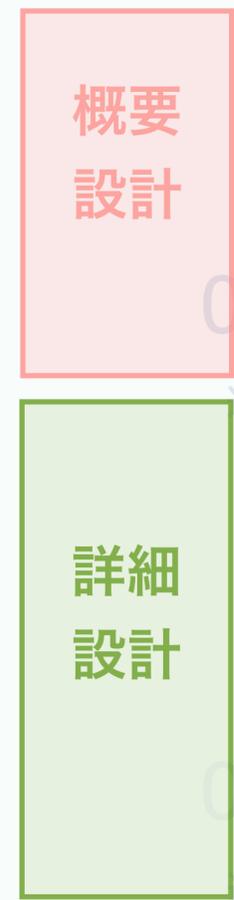
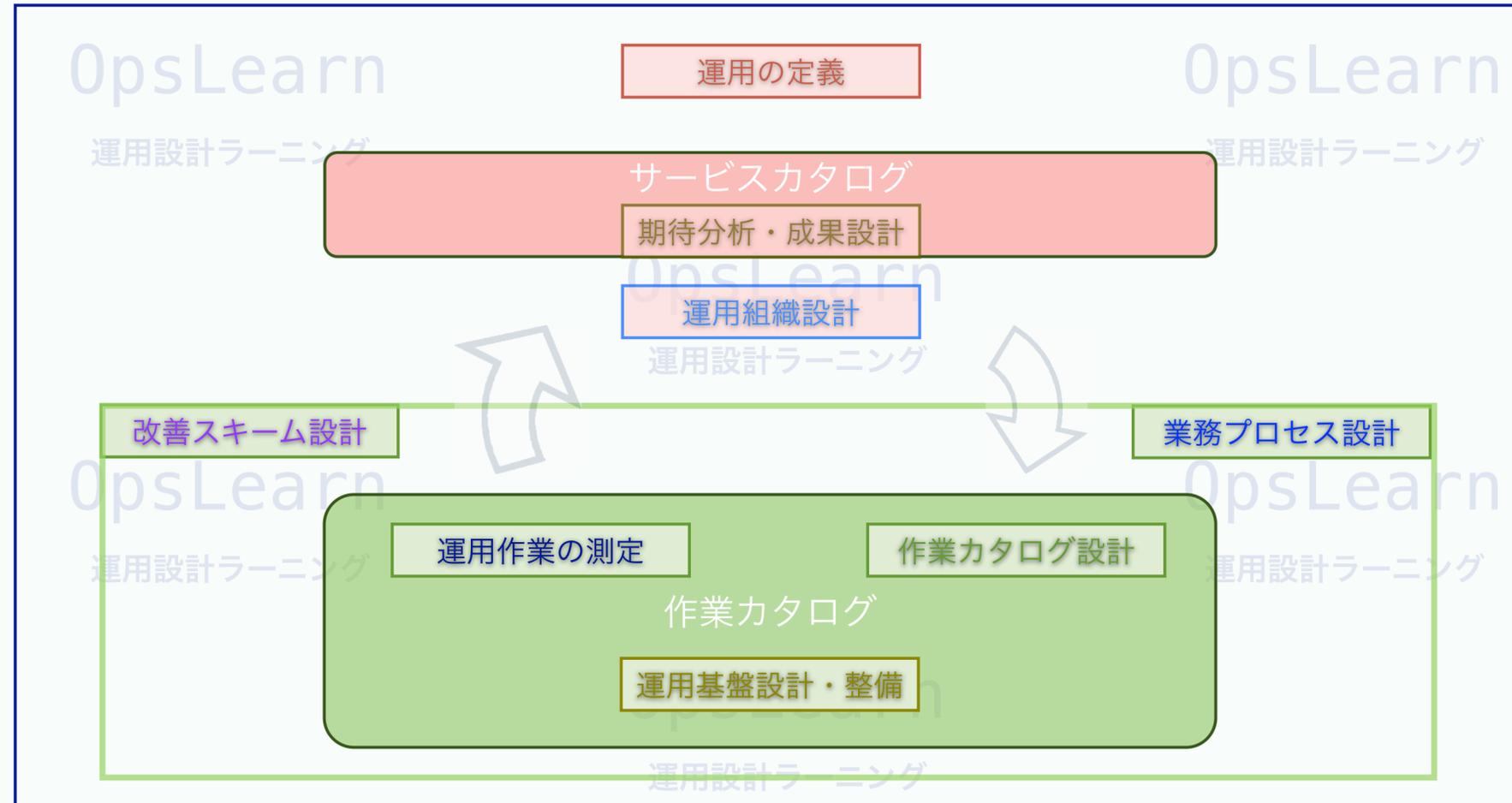
「楽な運用」の実現

▶ 運用に対する評価の適正化

適正な利潤を生む現場と、適切に評価される要員。

「稼ぐ運用」の実現

OpsLearnの全体像



本講義の位置付け

導入コース

- ・ 運用フレームワークの考え方
- ・ 運用現場の課題分析
- ・ 運用設計とは

運用フレームワークの考え方

運用設計ラーニング



OpsLearn

4つの視点

運用設計ラーニング

OpsLearn

3つの原則

運用設計ラーニング

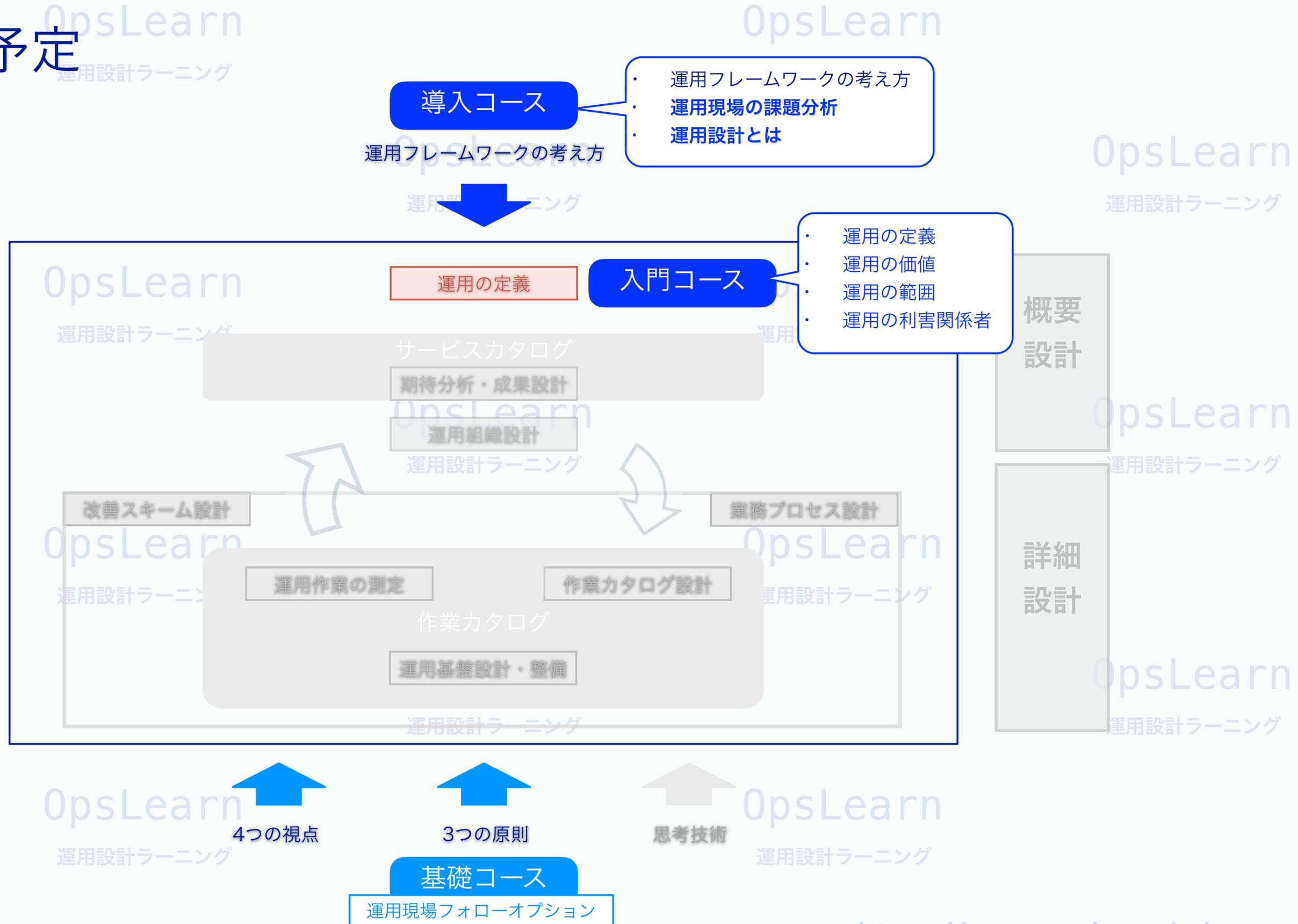
OpsLearn

思考技術

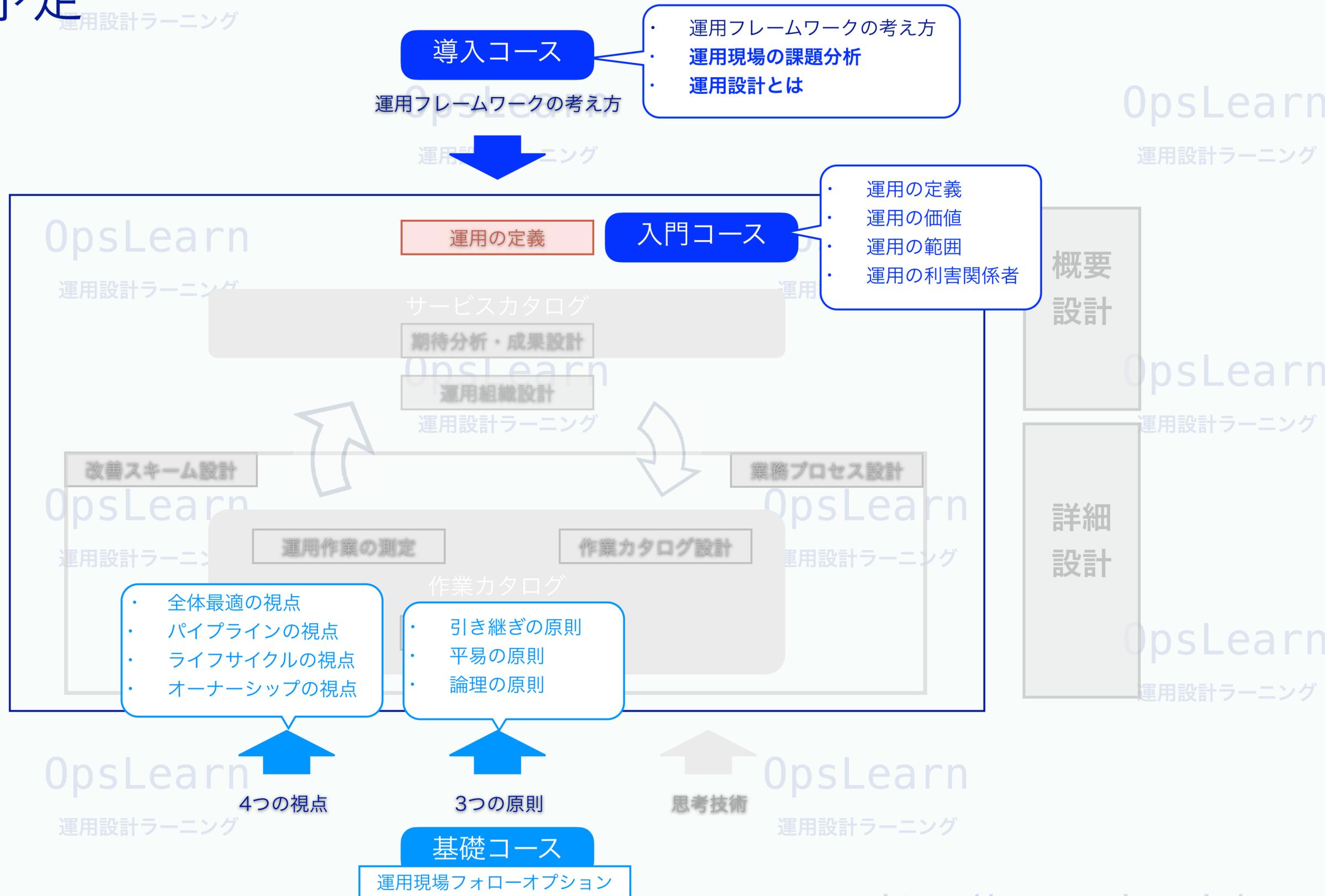
OpsLearn

運用設計ラーニング

参考: 今後の予定



参考: 今後の予定



運用設計導入コースの概要

「運用設計」について本格的に学ぶ前に
運用業務や運用組織を客観的に把握して最適化するために必要な
「運用フレームワーク」の考え方を身に付けていきましょう。

サービスやシステムの運用に関与する幅広い人々(マネージャー、エンジニア、セールスパーソン)向けのコースです。

講義1

今回

運用フレームワークの考え方

適切な運用設計を行うために、客観的かつ合理的に運用業務や運用組織を把握・整備する
「運用フレームワーク」という考え方について解説します。

講義2

12/22 (木)

運用課題の分析

第一回の「運用フレームワーク」の考え方に基づいて、
運用現場が抱える課題を分析していきます。

講義3

1/19 (木)

運用設計とは

第一回の「運用フレームワーク」の考え方に基づき、第二回の「運用課題の分析」で見えてきた運用課題の
根本原因を解消するための、あるべき「運用設計」について解説します。

運用設計導入コースの学習ポイント

導入コースは「**抽象**」的な内容が多くなります。

抽象

モデル

事物や表象を、**ある性質・共通性・本質に着目し**、それを抽（ひ）き出して把握すること。その際、他の不要な性質を排除する。

具体

実装(コード)

人間の感覚でとらえられるものであること。形や内容を備えていること。

空間的範囲

適用範囲が広い、適用対象が多い

適用範囲が狭い、適用対象が少ない

時間的範囲

寿命が長い、陳腐化が遅い

寿命が短い、陳腐化が早い

外部影響

影響を受けにくい、影響を吸収しやすい

影響を受けやすい、影響を吸収しにくい

実装との距離感

実装と遠い

実装に近い

課題の本当の意味での解決には「**抽象**」的な分析が必須となります。

IT業界でも「具体的な話」を望む方が多いですが、ITは本質的に「抽象化(モデル化)」することに価値の比重があり、適切に抽象化されたモデルを「各現場に合わせて最適に具体化(実装)」することではじめて大きな効果をもたらします。

(自分たちの前提条件や環境と合致しない「具体的な話」を実践することは、むしろ弊害を招くことが多いです。)

運用設計導入コースの講座の構成

本編の「抽象」的な内容を「運用あるある」や質疑応答で「具体」的に補完します。



「抽象」的な分析や議論に慣れていきましょう。

あらゆる設計・プログラミングでは極めて重要な能力ですが、できる人があまり多くないのが現実です。

今回の学習ポイント

インプット

本講義で意識してほしいこと

- ・ 根本解消できていない課題がある、それがなかなか解消できないのはなぜなのか。
- ・ 今までやってきた運用改善や自動化で、**持続的に効果を上げているもの**と、**そうではないものとの違い**はどこから生じているのか。

一次アウトプット

本講義から持ち帰ってほしい事

- ・ 「運用フレームワーク」という**考え方**
- ・ 「運用フレームワーク」が持つべき**性質**
- ・ 課題や物事を「**抽象化して考えること**」の**重要性**



- ・ 現在の**運用に足りないもの**は何か？ (視点、考え方など)
- ・ 自分達の「**運用フレームワーク**」に必要な**柱**は何か？

最終アウトプット

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

導入2

OpsLearn

OpsLearn

講師の経歴・運用経験

運用設計ラーニング

運用設計ラーニング

運用設計ラーニング

OpsLearn

運用設計ラーニング

講師の経歴:

20代: 大学から社会へ ～ 就職せずフリータに

士業のための資格勉強をしながら多様なアルバイトを経験。

結果的に、運用設計に必要な物の見方を獲得。

30歳: 黎明期のADSLベンチャーでネットワークエンジニア

業界未経験から経営破綻によるソフトバンク買収までの19ヶ月で多様な業務を経験

サポート対応、電話局作業、ATM運用、IPネットワーク運用、監視サーバ構築・運用、IPv6検証、運用自動化

30代半ば: 大手ASPで大規模運用

数千台の物理サーバの管理、数万ポイントの監視システムと監視センターの設計・構築・運用

数千万人の会員向けサービスのジョブ基盤の設計・構築・運用

講師の経歴: 大学から社会へ: 就職せずフリータに

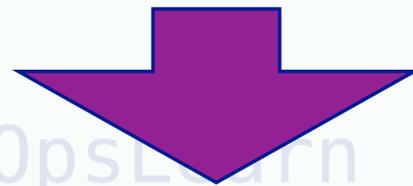
日本大学法学部 経営法学科 卒業

アルバイト経験

・ほとんど大学行かず、ゼミにも参加せず

- ・ 学位論文を書いたことがない
- ・ 法律には興味あったので**行政書士は合格**
 - ・ 司法書士は勉強不足で受験せず

- ・ ファーストフード
- ・ 什器運搬・設置



・就職せずにフリーター

- ・ 税理士や不動産鑑定士を受けると称して勉強せず...
- ・ PCには興味あったので**マシンの自作**
 - ・ 自主勉強会で学んで二種情報処理を取得 (一種は不合格)

- ・ スーパー品出し
- ・ 配膳人
- ・ ホテル清掃アルバイト
- ・ 新聞奨学生
- ・ テープ起こし
- ・ TVエキストラ (主要キー局制覇)
- ・ 大型郵便局清掃スタッフ
- ・ カスタマーセンタースタッフ
- ・ データ配信事業者アルバイト
- ・ 派遣事業給与計算スタッフ
- ・ 校正オペレーター

アルバイト経験がのちのち運用設計に効いてきた

講師の経歴: フリータ時代のおわり ~ 転機はアウトプットから

とある技術ML

- ・ **モバイル機器でサーバを立てる話題**
 - ・ さっそく何台か調達して質問しまくる&グローバル接続
- ・ **MLで教わったことをまとめてWebで公開**
 - ・ いつか「雑誌で紹介されそうな程度」の手応え

個人がグローバルIPアドレスを取得できるようになった

**グローバルIPアドレスを取得した日から3日3晩
いつ寝たのか覚えてないくらいサーバ構築に熱中**

ネームサーバ、メールサーバ、Webサーバ



- ・ **アスキーの編集長から雑誌の記事執筆依頼**
 - ・ 「雑誌にサイト紹介」はされずにいきなり記事著者に
- ・ **記事執筆を履歴書に書いて就職活動**
 - ・ I社は書類選考で落ちて、東京めたりっく通信は面接合格



東京めたりっく通信入社

東京で最初に商用ADSLサービスを開始した会社

大阪は「大阪めたりっく通信」名古屋は「名古屋めたりっく通信」

講師の運用経験: ADSLベンチャー時代 (1年7ヶ月)

「運用チーム」に配属

- ・とにかく何でもやらなければいけなかった
 - ・未熟でも、未経験でも何でもやらせてもらった。
- ・自律的に動いていかないと業務が回らない
 - ・人数が少なく「誰かが何かをする」を待つ時間が無い。

電話局作業、ATM運用、IPネットワーク運用
いずれも未経験ながら、並行して担当していた。

興味があった監視サーバ構築・運用、運用自動化
に積極的に絡み、担当にしてもらった。



- ・社会人9ヶ月目で**経営危機**
 - ・派遣会社の人員が引き上げて少数の社員だけで業務。
- ・社会人19ヶ月目で**会社消滅**
 - ・運用現場も消滅した。

2人体制で12時間2交代勤務 (1週間交代)
若かったけど、しんどかった。。

退職で引き継ぎが必要になったため、
自動化した業務の**全面手動化**を行なった。

運用現場は永続とは限らない。

「稼げない運用」は自分たちの首を締める。

講師の運用経験: 大手ASPで大規模運用 (最初の1年半)

「大規模運用」がやりたくて転職

・ 想像以上の「大規模運用」だった

- ・ ミーティングも多く、定時後からが自分の仕事時間。

・ 影響範囲が大きく、**想定外事象**も発生。

- ・ 「ワールドワイドで初の事象」と頻繁に言われる。



・ とにかく自動化していった

- ・ 当時はエンジニアでも「手でやれば良い」が多数派

・ 「スケールする、保守できる自動化」を意識

- ・ 引き継ぎできない自動化の辛さをすでに認識。

物理サーバー**数千台** (毎月100台程度増える)

10人(5チーム)で構築・運用管理 (各チーム 数百~1千台程度)

・ サーバ自動管理台帳 (3000台以上、毎日更新) [2004]

・ メンテナンスリアルタイムボード (100人超参加) [2004]

・ サーバ構築・監視・レポートスクリプトなど

講師の運用経験: 大手ASPで基盤設計・構築・運用 (次の3年半)

「大規模運用」で運用改善・自動化したけれど

- ・ 「運用改善」しても「自動化」しても、先が見えてこない
 - ・ たしかに良くなっているが、根本解決した感じがしない。
- ・ 改善すればするほど、より大きな課題に直面して、ゴールが遠くなる
 - ・ 「課題解決」という蜃気楼を追う気分。



ジョブと監視のプラットフォームを正社員一人で設計・構築・運用。

- ・ 「運用改善」や「自動化」の限界に気付いた
 - ・ 手段としては有効だが、抜本的に良くはならない。
- ・ 「抜本的な変化」を意識
 - ・ 少なくとも3年、できれば7年持続する変化を目指した。

周囲は満足しているし、数字も出ているし、評価も上がった。

(退職後10年の現在も、当時の成果を多くの元同僚が覚えてくれている)



(事例が記事に)

明らかに破綻しているジョブ基盤

ジョブ基盤の影響範囲 **数千万人の会員**

仕様が不明のジョブクラスタと、スケールしないジョブマネージャ

明らかに破綻している監視基盤

監視ポイント **数万ポイント**

アラート通知メールが、年間数百万通 (1日1万通以上)

明らかに破綻している監視センター

障害対応シート **数十万セルのExcelワークシート**

増強時の転記ミスで対応ミスが頻発。

講師の運用経験: 最終的に行き詰まる

周囲は満足しているし、数字も出ているし、評価も上がっている。

しかし、山を超えたら、次の山が来ることの繰り返し

- ・ 課題解決してるはずなのに、先が見えてこない
 - ・ 個々にはちゃんと解決しているが、**根本解決**した感じがしない。
- ・ 改善すればするほど、更に大きな課題に直面して、ゴールが遠くなる
 - ・ 「課題解決」という蜃気楼を追う気分。

「なぜうちの運用はこんなにしんどいのか。他所はどうしているのか」と悩む。

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

導入2

OpsLearn

OpsLearn

講師の経歴・運用経験

運用設計ラーニング

運用設計ラーニング

運用設計ラーニング

OpsLearn

運用設計ラーニング

講師の運用研究活動

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

参考: ITILに期待した時期

2008年 ITSMFに自腹で参加

運用設計ラーニング

参加費2万5千円、公式本は8万円くらい

当日話します

当時は、v2からv3への移行期

講演で唯一、マイクロソフトの情シスが「サービスカタログ」でグダグダ運用から脱却した話は、とても面白くて参考になりました。

公式本で、一通り学んで得られたのは「サービスカタログの概念」と「ライフサイクルという考え方」の2つでした。これらは今でも生きています。

肝心の「サービスオペレーション」が一番ボリュームが薄く、極端に抽象度が高いか極端に具体性が高いかの両極端になっていて、その間を繋ぐ概念が無いので、テキストだけでは大規模運用の実務では使えなかった。

ITIL v3からは離れて、独自調査を進める

運用研究を始める

2009年7月 JANOG(ネットワークオペレータコミュニティ)で初発表

「自分はこんなことを考えている。うまくやっている人の話を聞きたい。」という発表。

登壇直後に一番最初に喰い付いてきたのは通信会社最大手だった。

2009年11月 JPNIC主催 Internet Weekで「運用方法論」を発表

IW史上、最もアンケートコメントが多いセッションとして非常に好評。

通信キャリア、新聞社、金融機関の運用部門関係者からも高い評価を得る。

2010年度 Telecom ISACと共同研究

NTT東日本、NTT西日本、NTTコミュニケーションズ、KDDI

ソフトバンクテレコム、ソフトバンクモバイル、富士通、e-Accessの八社と1年間の共同研究

「運用組織モデル」の確立をリードし、運用研究に自信を深める。

運用研究の初期にわかったこと

「なぜうちの運用はこんなにしんどいのか。他所はどうしているのか」
とみんなが悩んでいた。

通信キャリア、新聞社、金融機関の運用部門関係者も悩んでいた。

NTT東日本、NTT西日本、NTTコミュニケーションズ、KDDI
ソフトバンクテレコム、ソフトバンクモバイル、富士通、e-Accessの八社も悩んでいた。

管理会計、生産工学、システム工学領域などに知見が無いか調査

「運用」そのものの研究分野は存在しなかった。
システム工学が近いものらしいが、運用現場で使えるものではなかった。

「運用」に関する先行研究は、一部の研究会論文を除きほぼ存在しなかった。

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

導入3

OpsLearn

OpsLearn

運用あるある

運用設計ラーニング

運用設計ラーニング

運用設計ラーニング

OpsLearn

運用設計ラーニング

講師も数多くの「課題」に直面

リソース間の依存関係が複雑で、影響がどこに出るかわからない。

停止や更新実施時に、構成情報上では影響の無いはずのリソースに影響が発生する。
スクリプトや手順書の改修により、想定していなかったリソースでエラーや事故が発生する。

想定外(質的・量的の双方)が日常的に起こる。

運用リソースを、想定外の用法で使われる、使おうとされる。全く想定外の運用相談を受ける。
何かのきっかけ(ビジネス起因、障害起因)で、想定外の量の依頼や監視アラートが来る。

メリットとデメリットのトレードオフが巨大化していく。

提供している基盤が好評で、利用者側の生産性が向上する、一方で運用する側の受け入れ・運用負荷は急増する。
運用業務の自動化により、平時の生産性は高くなるが、不具合が発生すると今までのプラスを吹き消すくらいのマイナスになることがある。

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

導入3

OpsLearn

OpsLearn

運用あるある

運用設計ラーニング

運用設計ラーニング

運用設計ラーニング

OpsLearn **「課題」への対応 (講師の経験を例に)** OpsLearn

運用設計ラーニング

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

「課題」への対応: 「課題」が発生

「課題」が発生すれば、当然に対応する



「課題」への対応: 「運用改善」や「自動化」で課題解消

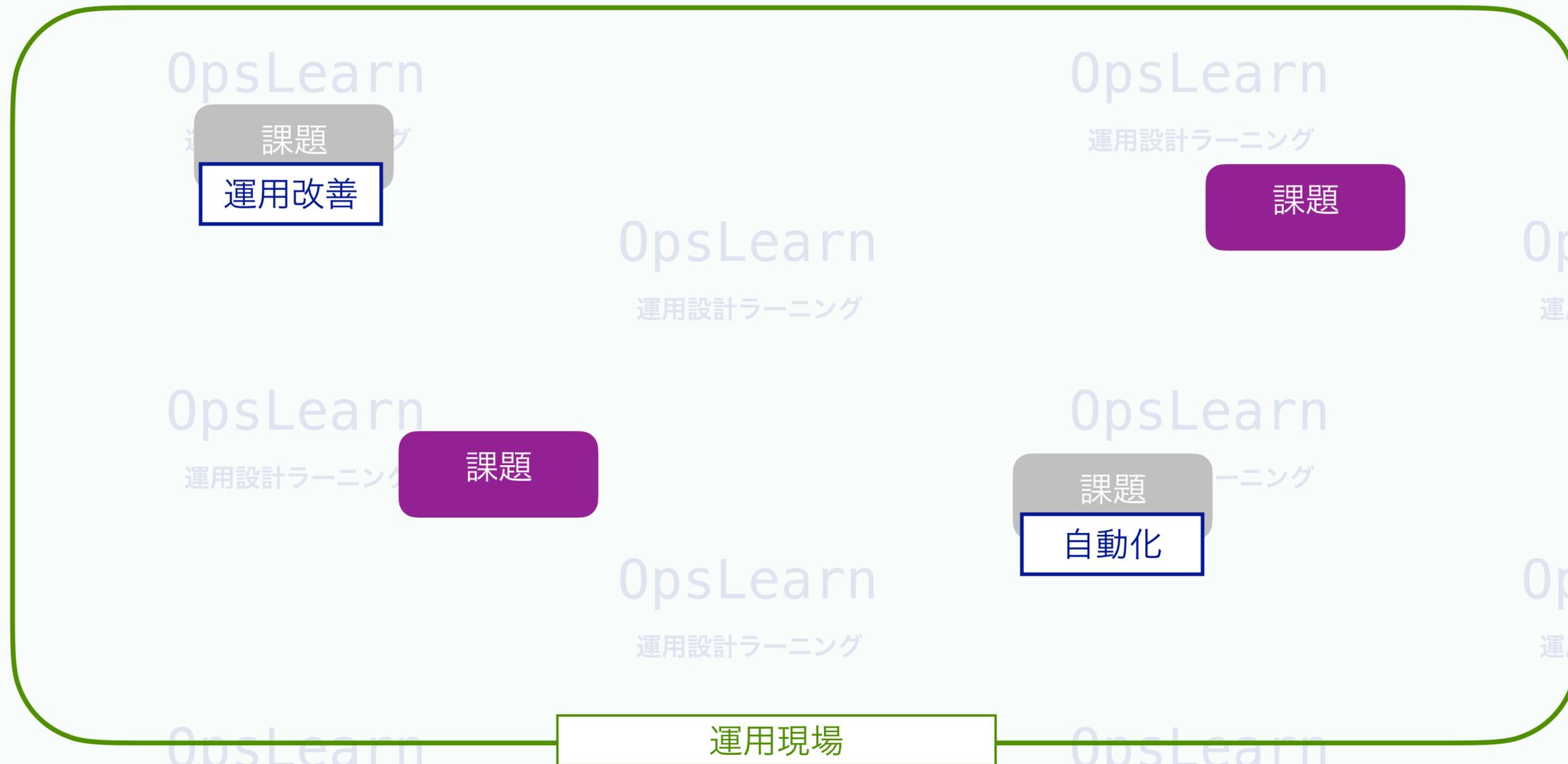
「運用改善」や「自動化」で課題が解消した (ように見える)



運用改善や自動化をした人は、この時点で評価される。

「課題」への対応: 次々と「課題」が発生

次の「課題」が発生すれば、当然に対応する



「課題」への対応: 追加の「運用改善」や「自動化」で課題解消

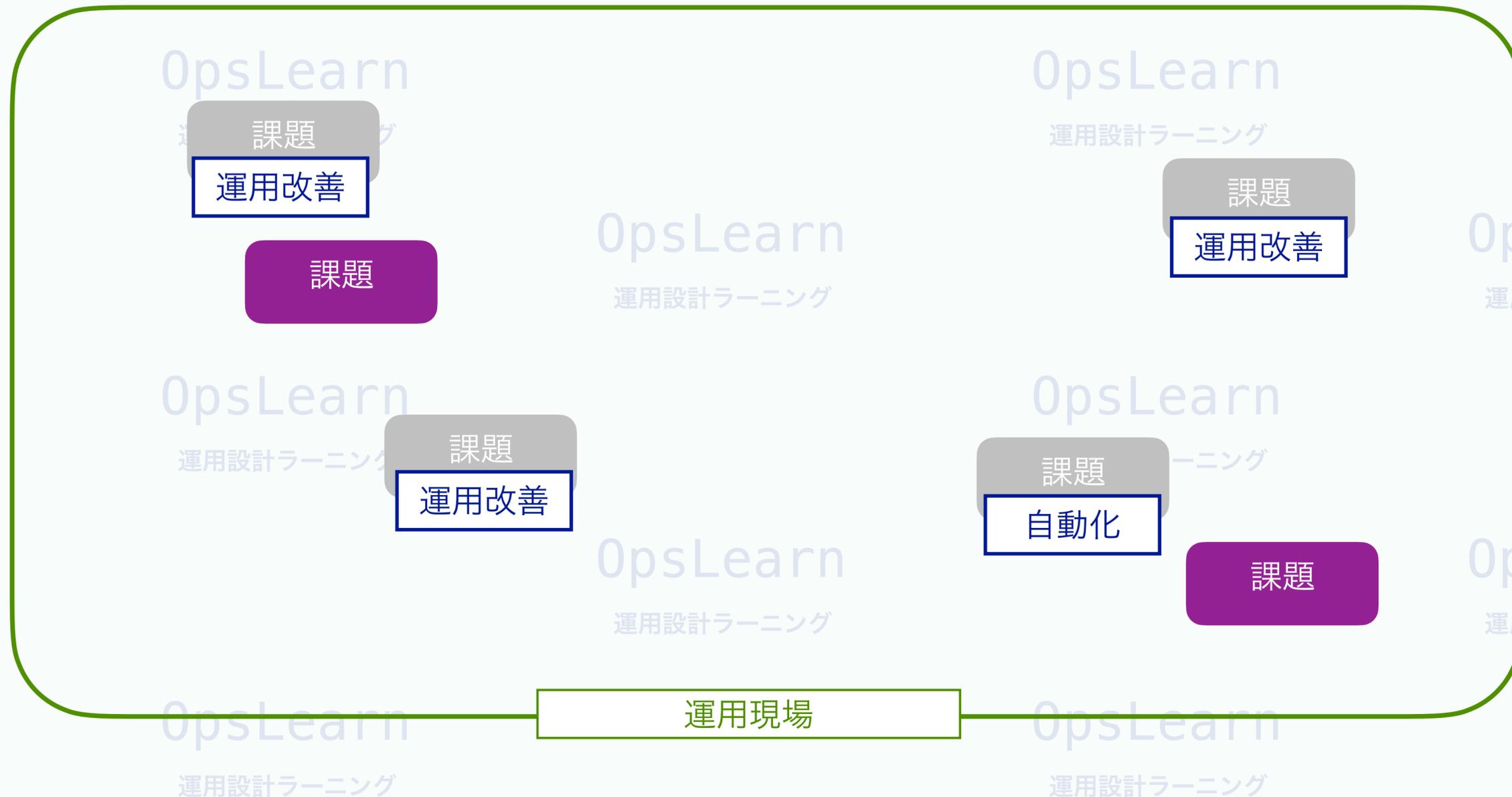
「運用改善」や「自動化」で課題が解消した (ように見える)



運用改善や自動化をした人は、この時点で評価される。

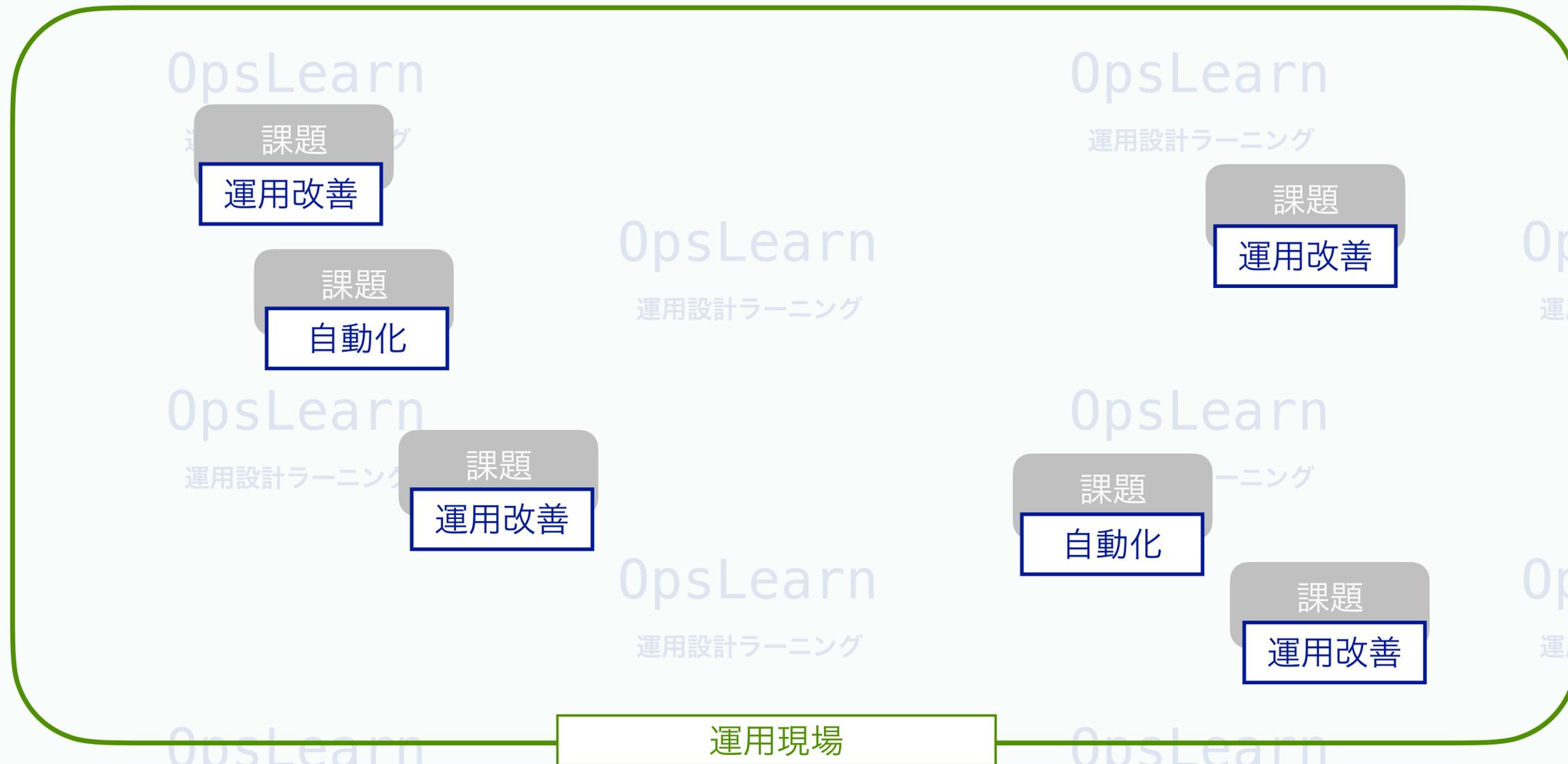
「課題」への対応: 改善した隣接領域で更に「課題」が発生

更に「課題」が発生すれば、当然に対応する



「課題」への対応: 更に「運用改善」や「自動化」で課題解消

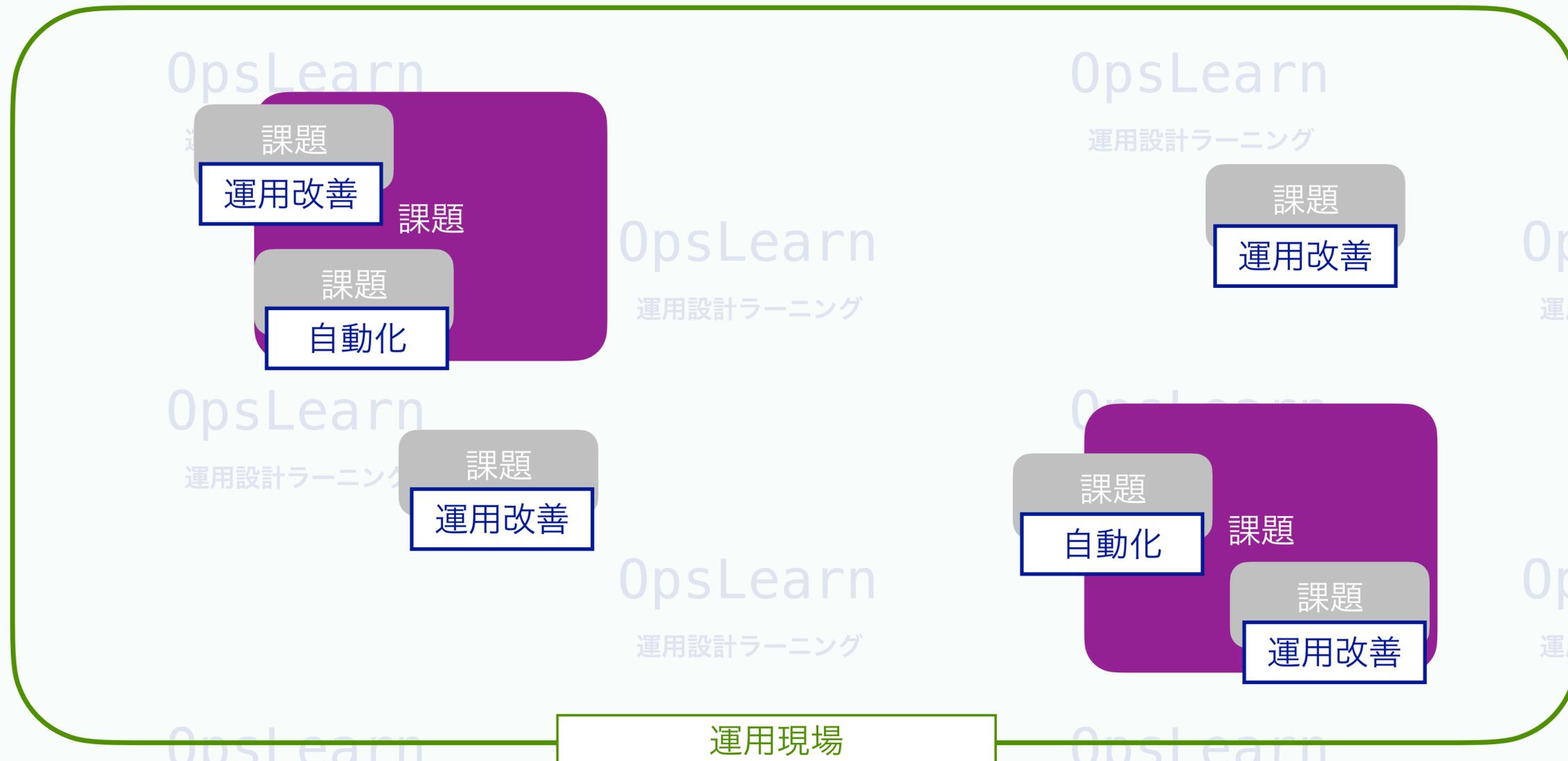
「運用改善」や「自動化」で課題が解消した (ように見える)



運用改善や自動化をした人は、この時点で評価される。

「課題」への対応: 複合的に「課題」が発生

「運用改善」や「自動化」で解消した課題を含めた**複合的な「課題」が発生**

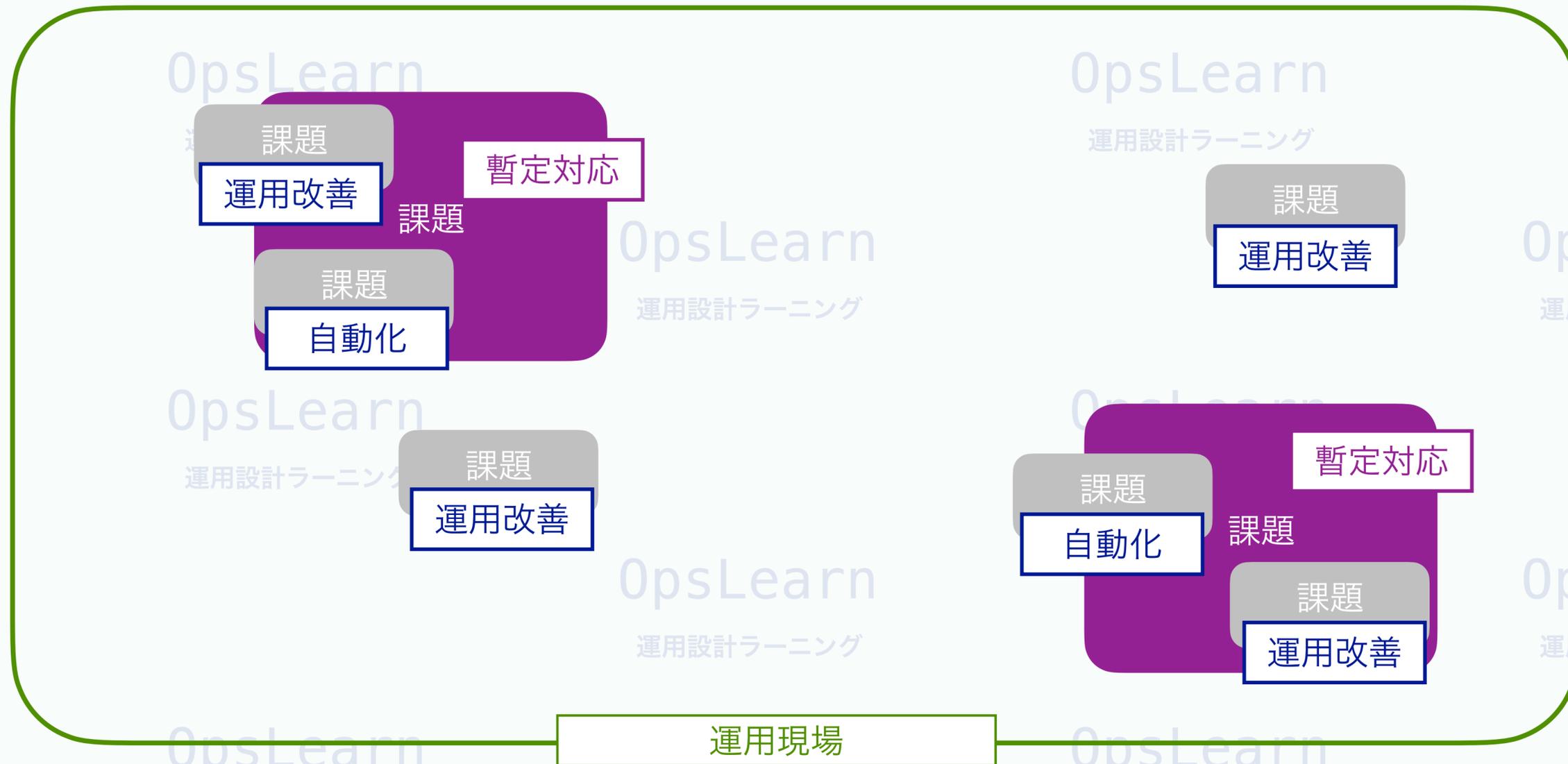


ルールや自動化の間で**矛盾や衝突(コンフリクト)**が発生する、**自動化により業務が硬直化する**、など。

過去に評価された「運用改善」や「自動化」に問題があった可能性が判明する。

「課題」への対応: 複合的に「課題」に対する「暫定対応」の発生

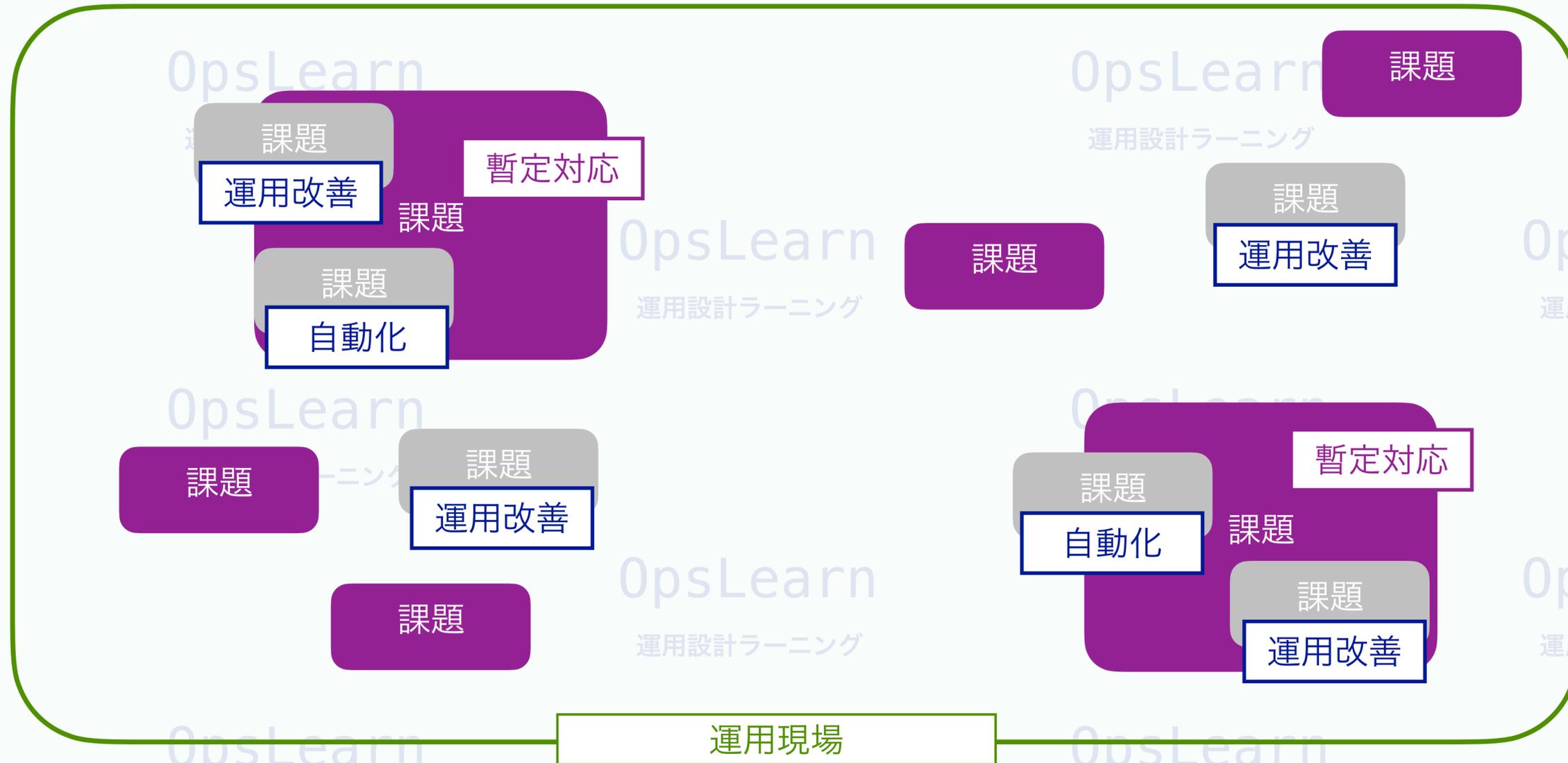
根本解決の検討には時間と工数が掛かるため、**暫定の対応**を実施



課題が残ったまま、「暫定対応」という例外的な処理を行うため、**生産性が低下**する

「課題」への対応: 更に「課題」が発生

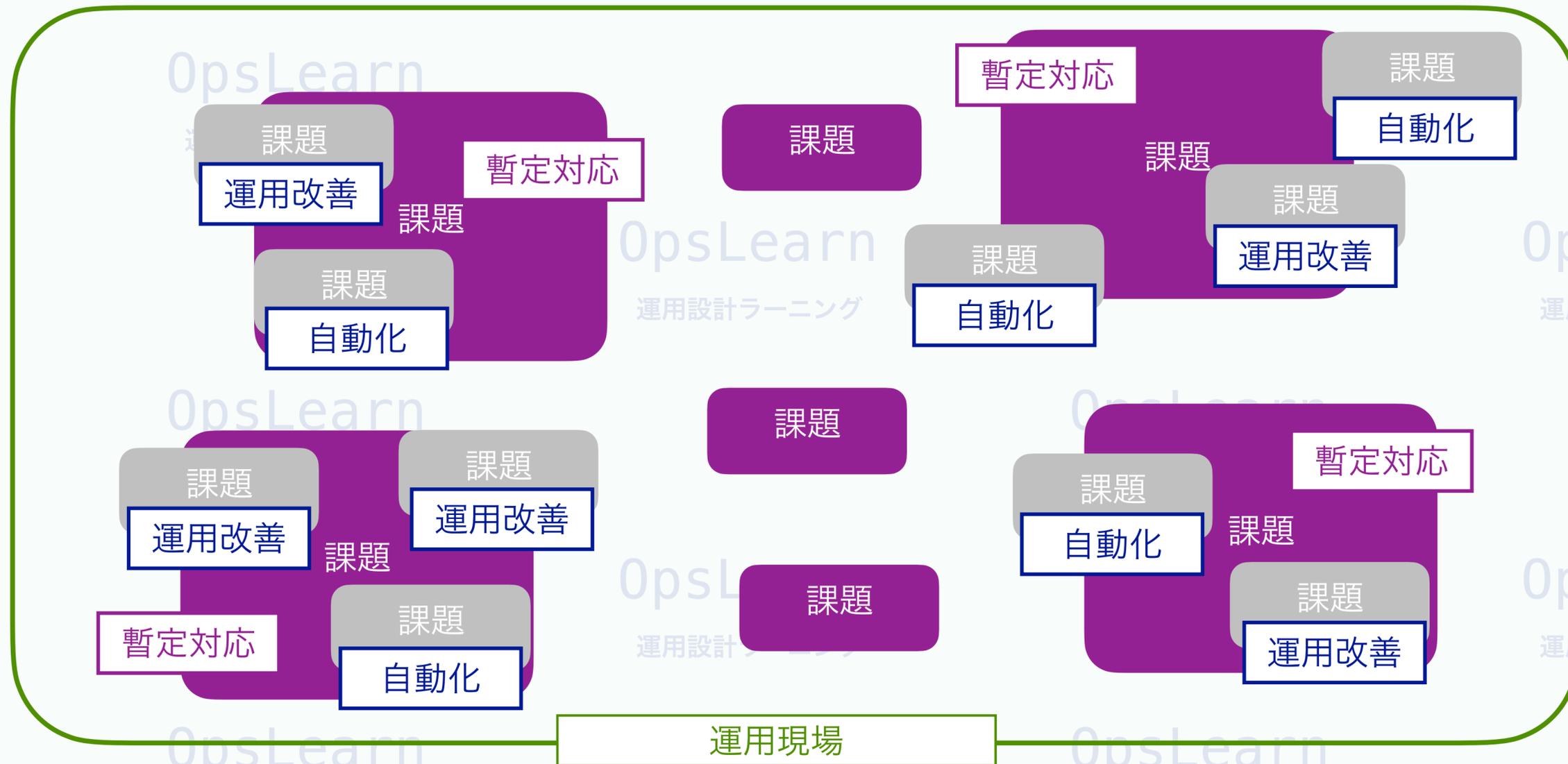
暫定対応に追加して、課題への対応が発生する



「暫定対応」という例外的な処理 + 課題対応を行うため、生産性が更に低下する

「課題」への対応: 更に「課題」が発生、を繰り返す

運用現場のリソースが不足し、恒常的に課題への対応ができなくなる



現状維持が精一杯になり、新しい業務への対応や提案をする余裕が失われる。

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

導入3

OpsLearn

OpsLearn

運用あるある

運用設計ラーニング

運用設計ラーニング

運用設計ラーニング

講師が「運用あるある」で気付いたこと

運用設計ラーニング

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

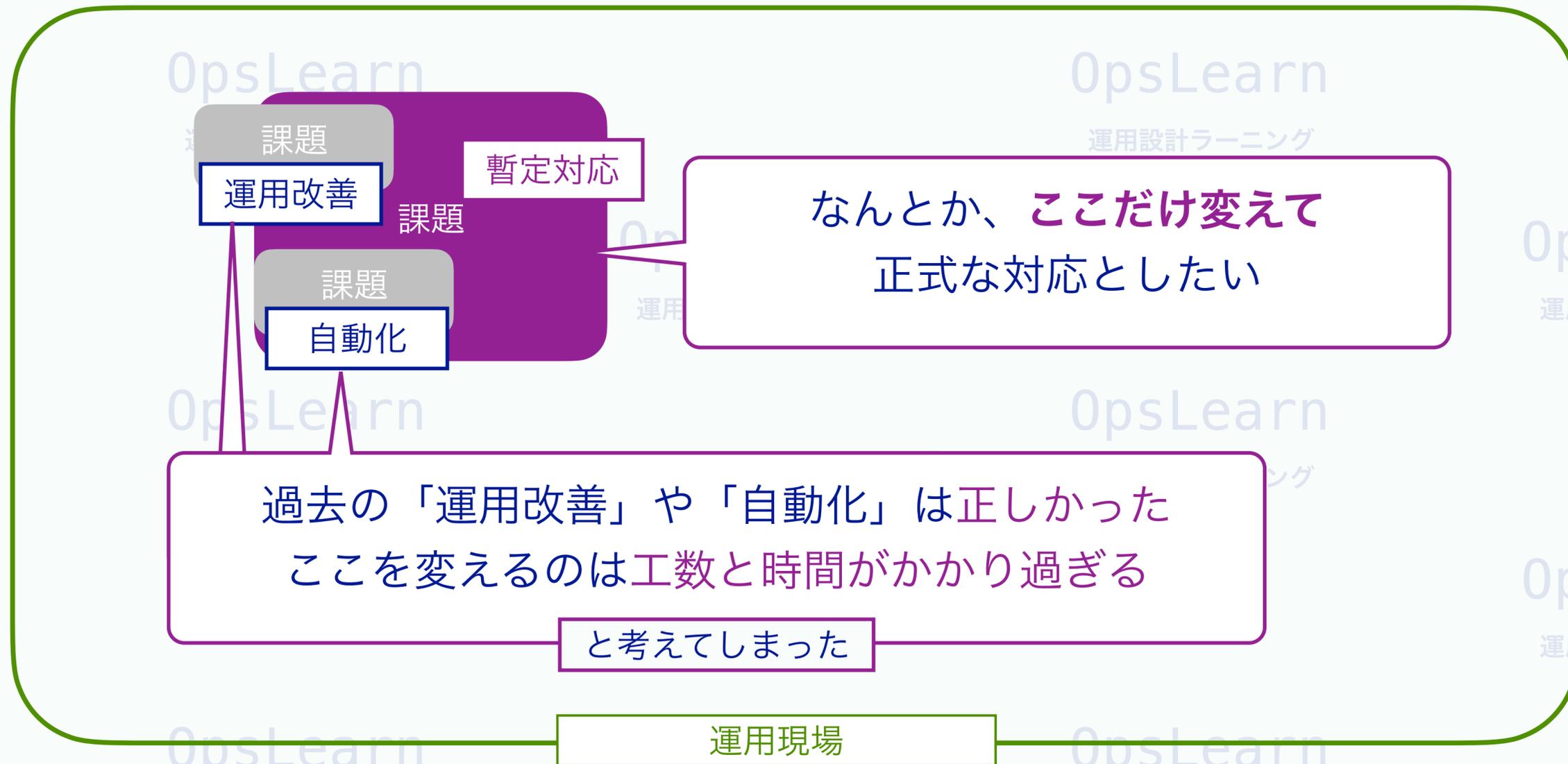
OpsLearn

運用設計ラーニング

講師が「運用あるある」で気付いたこと

問題点2: 過去の延長線上で課題を解決しようとしていた

過去の「運用改善」や「自動化」や、現在の前提条件が正しいと考えた

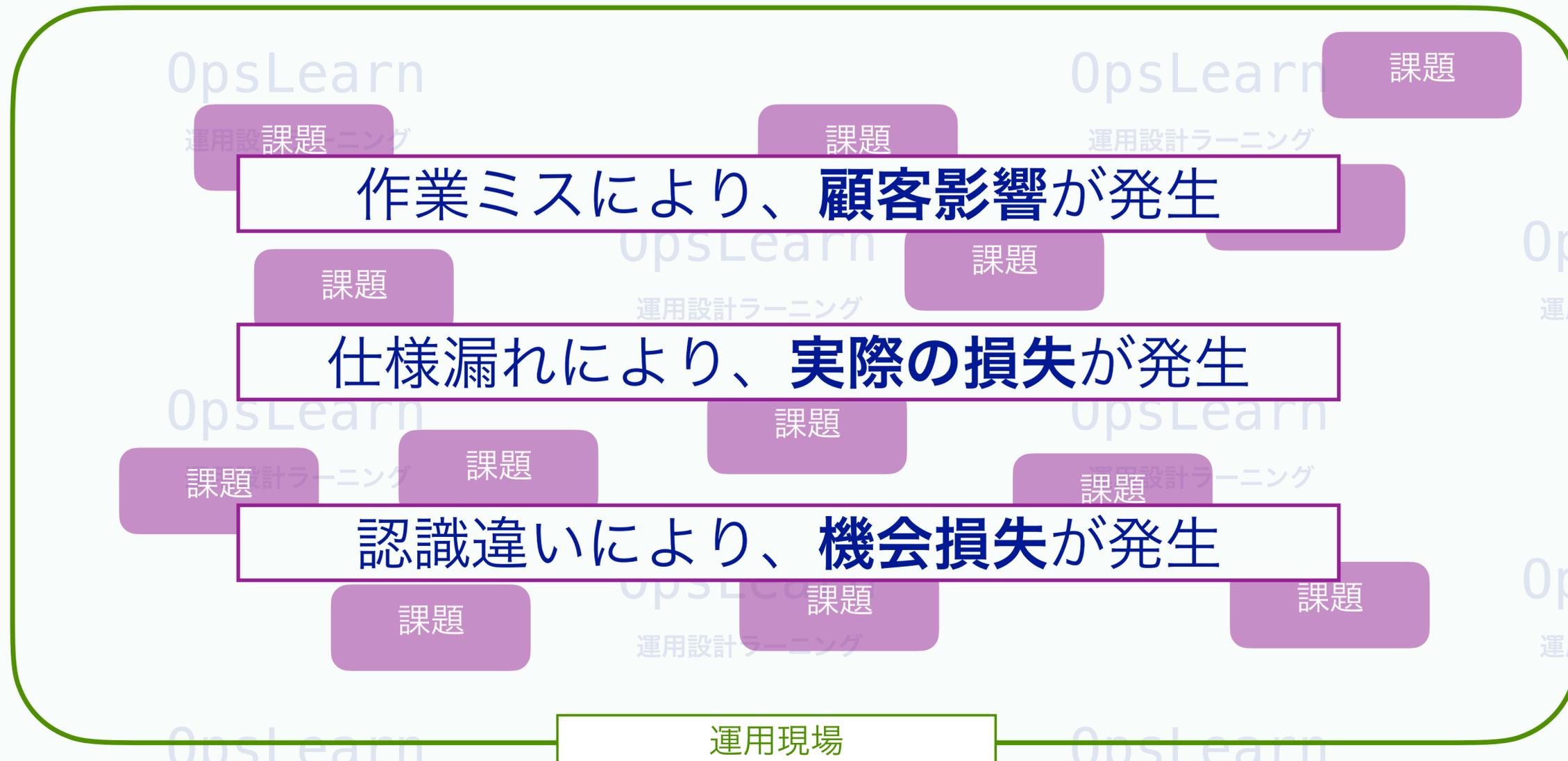


「前例主義的」で、抜本的解決の視点に欠けていた。

講師が「運用あるある」で気付いたこと

問題点3: 全ての「課題」が発生後の対応になっていた

何かが起きてから課題に気付き、報告しながら緊急対応していた



「受け身的」で、課題の発生自体を抑制する視点に欠けていた。

講師が「運用あるある」で気付いたこと

課題への対応における3つの問題点

問題点1: 「課題」への対応を「点」に対して行なっていた

「場当たりの」で、長期的な視野に欠けていた。
「目の前」に見える範囲で、個別的に課題解決しようとしてしまった。

問題点2: 過去の延長線上で課題を解決しようとしていた

「前例主義的」で、抜本的解決の視点に欠けていた。
歴史的経緯を温存し、現在の状態や条件を前提として課題解決を図ろうとしてしまった。

問題点3: 全ての「課題」が発生後の対応になっていた

「受け身的」で、課題の発生自体を抑制する視点に欠けていた。
「課題は発生するもの」という慣れが生じ、そもそも「課題を発生させない」という視点が無かった。

講師が「運用あるある」で気付いたこと

どうすべきなのか？

問題点1: 「課題」への対応を「点」に対して行なっていた

「課題」に対して(可能な限り)面に対応をするべき

問題点2: 過去の延長線上で課題を解決しようとしていた

「課題」を(可能な限り)根本的に解決するべき。

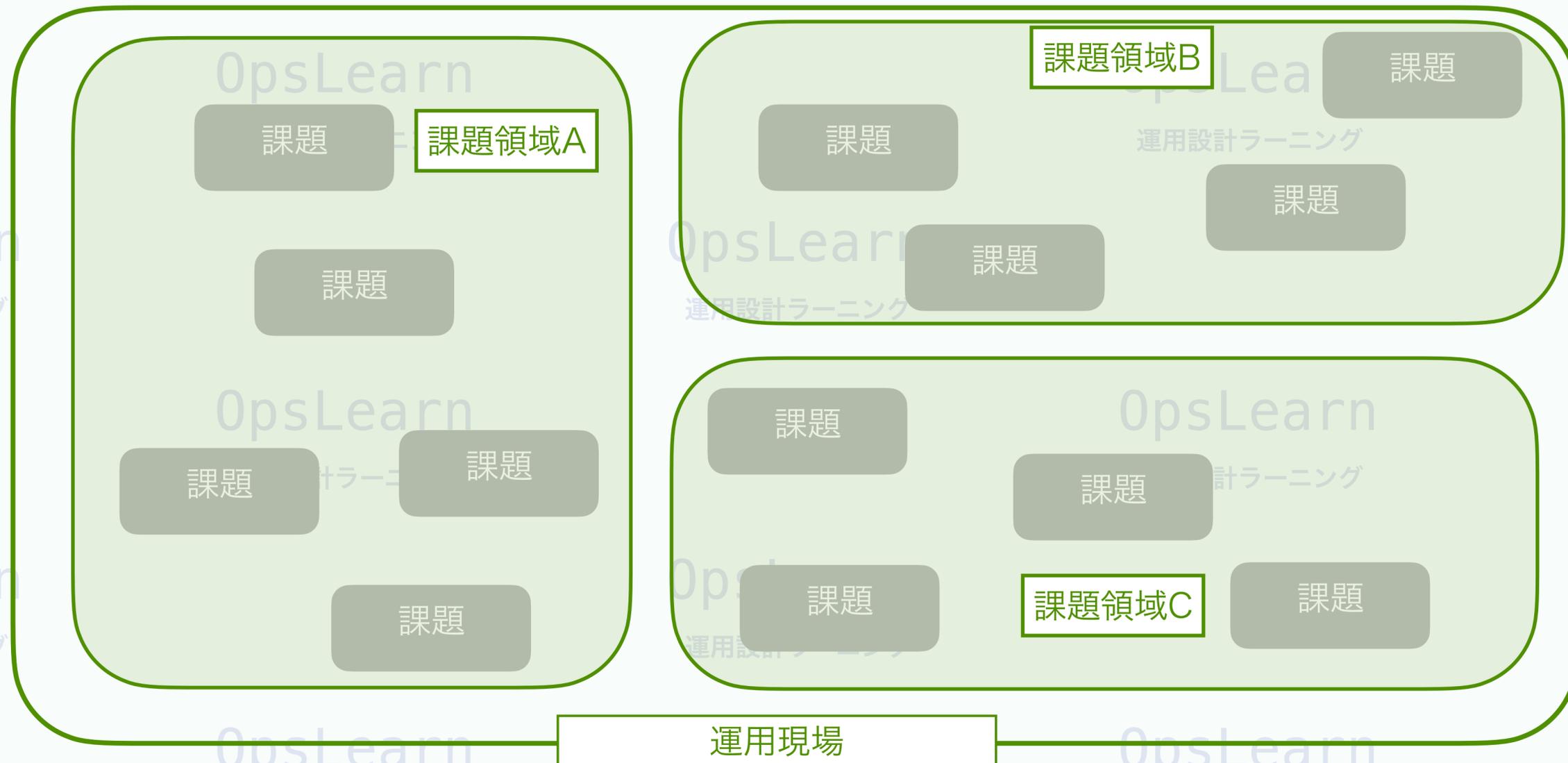
問題点3: 全ての「課題」が発生後の対応になっていた

「課題」を(可能な限り)発生させずに、予防的に対応すべき。

講師が「運用あるある」で気付いたこと

対処1: 課題解決を「面」で考えるようにした

「課題」を個別の「点」で捉えるのではなく、「面」で捉える。



しかし、「面」をどう捉えるべきなのかがわからなかった

講師が「運用あるある」で気付いたこと

対処1: 課題解決を「面」で考えるようにした

「課題」を個別の「点」で捉えるのではなく、「面」で捉える。

Step1. 本当の「設計」をしていないことに気付いた

自分の言っていた「設計」は「実装を説明しているもの」でしかなかった。

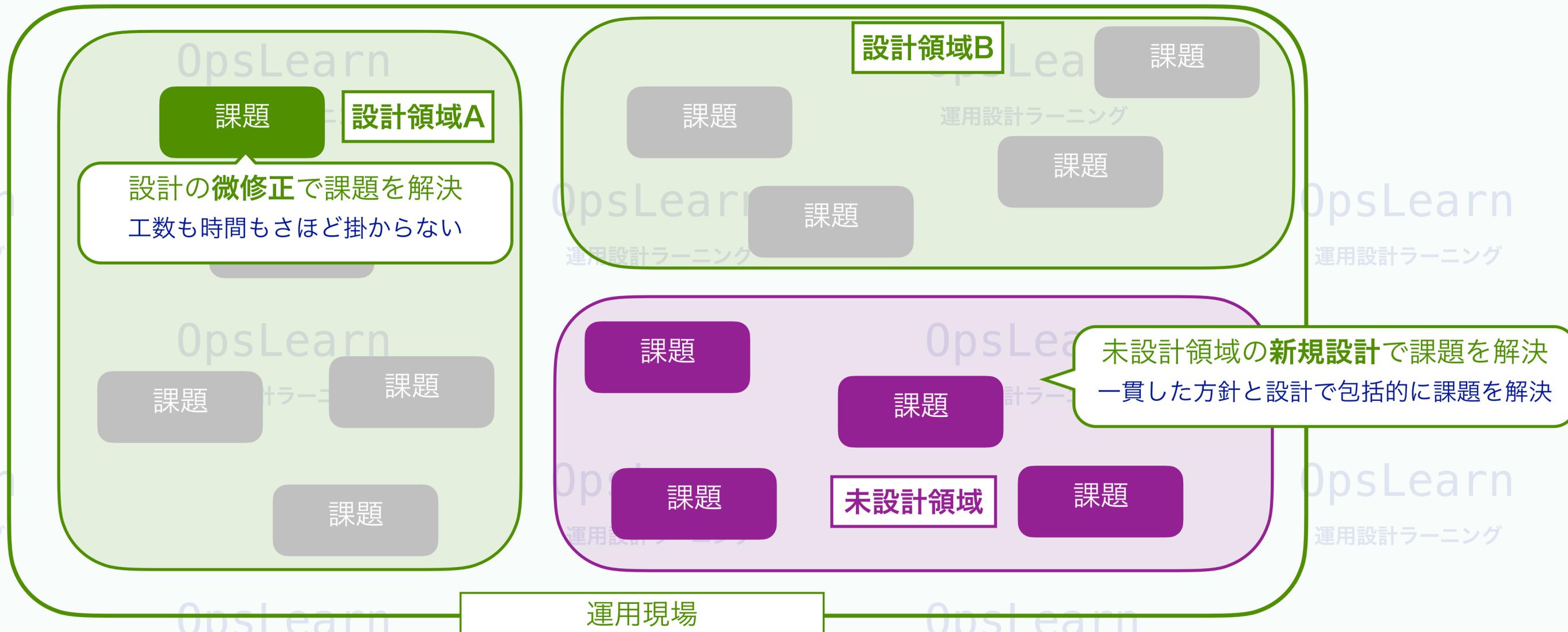
そこには「設計思想」と言えるものは存在せず、他人からのコピーや「実装の書き起し」でしかなかった。

「設計がない」から「課題を面で捉えられない」

講師が「運用あるある」で気付いたこと

参考: 本当の「設計」と「課題」

設計がされている領域は、設計の「修正」で課題に対応できる。

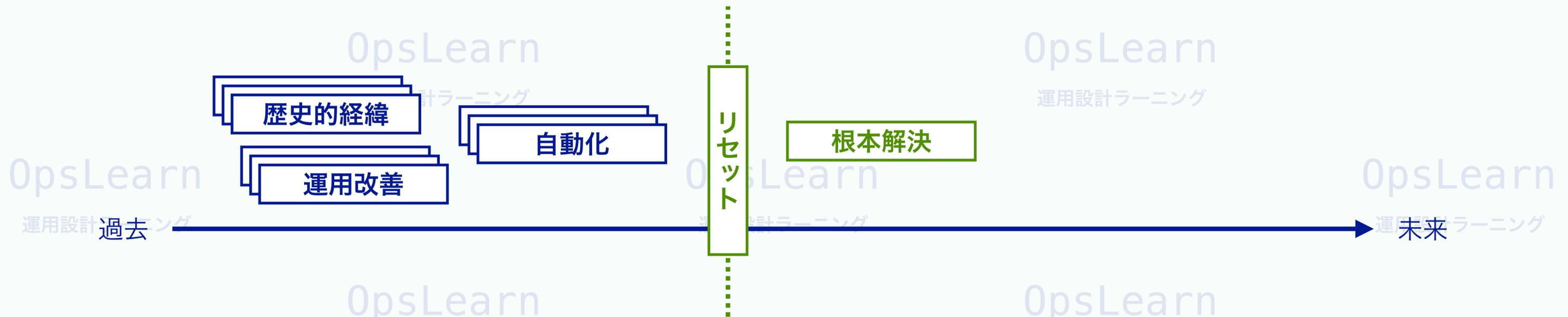


設計がされていない領域は、課題を「設計で解決」(根本解決)する。

講師が「運用あるある」で気付いたこと

対処2: 課題を「根本解決」で考えるようにした

「課題」を今の延長線上に捉えるのではなく、「非連続的」に考える。



「歴史的経緯」を一旦リセットする

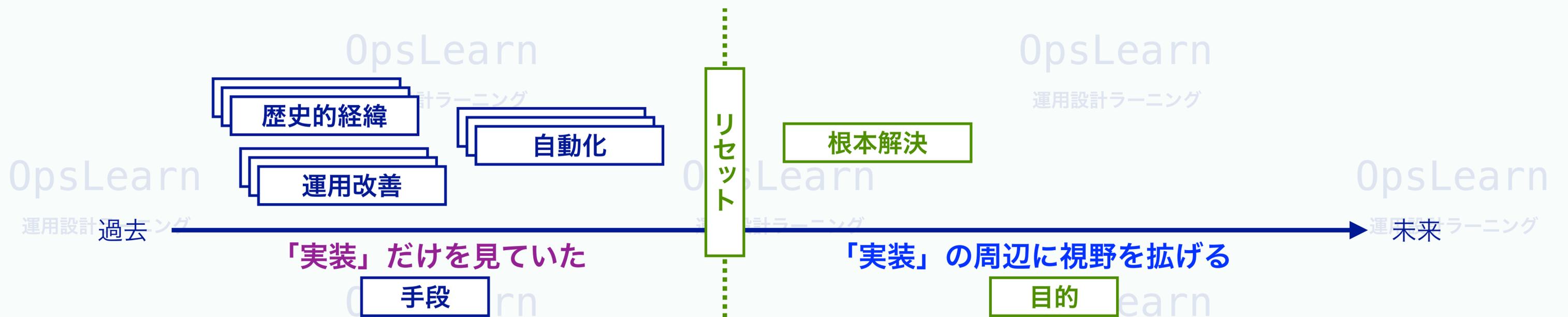
過去の「運用改善」や「自動化」の成果を一旦忘れる

しかし、何をすれば「根本解決」になるのかがわからない。

講師が「運用あるある」で気付いたこと

対処2: 課題を「根本解決」で考えるようにした

「課題」を今の延長線上に捉えるのではなく、「非連続的」に考える。



Step2. 「設計」の対象の捉え方が狭すぎることに気付いた

設計の対象を「実装」そのものに限定して考えてしまっていた。

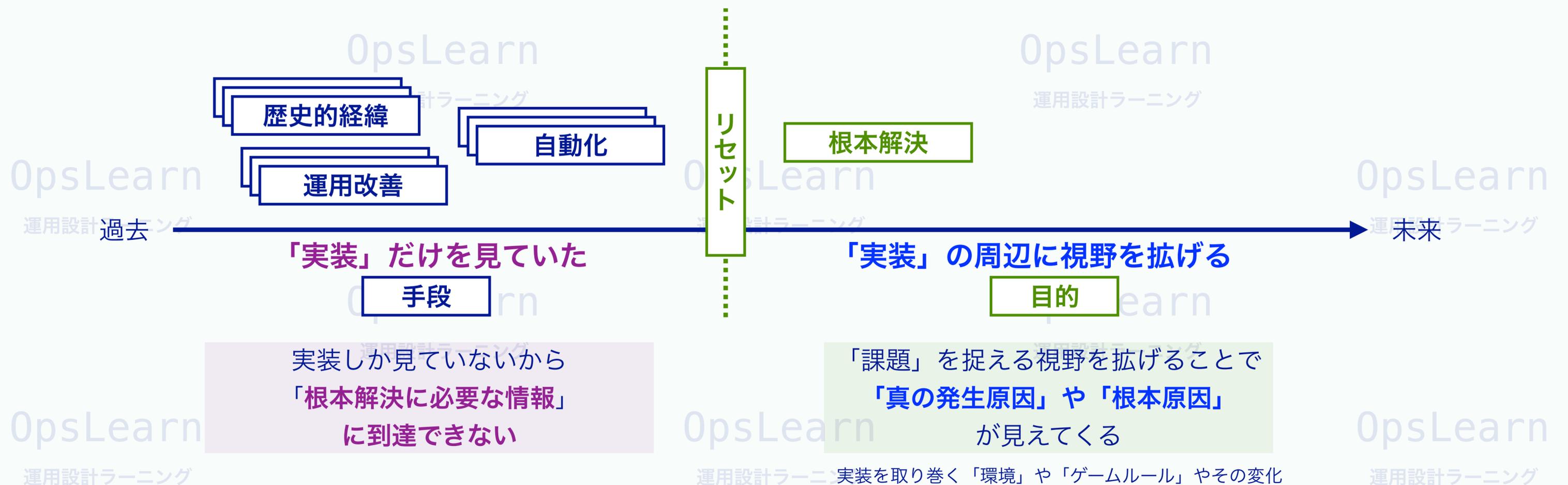
「実装」を取り巻く環境やゲームルール(ビジネスそのもの)まで見ていなかった。

「実装しか見ていない」から「今の延長線上でしか捉えられない」

講師が「運用あるある」で気付いたこと

参考: 「課題」と本当の「根本解決」

課題の**真の発生原因**や**根本原因**は、**実装の外部**にあることが多い。



大規模運用での経験

「実装」ですぐやれる事はほぼやり切っていたため、周辺に視野を広げざるを得なかった。

(現状はギリギリこなせても、大幅に増えることが確実だったため、いずれ運用破綻することが目に見えていた。)

講師が「運用あるある」で気付いたこと

対処3: 課題発生の「予防」を考えるようにした

「課題」に事後的に対応するのではなく、「**予防**」を考える。

事後的対応となる課題

課題の発生を予測できない

課題への対応を準備できない

課題の発生を予防できない

減らしたい

完全に無くすことはできない

事前的対応できる課題

課題の発生を予測できる

課題への対応を準備できる

課題の発生を予防できる

増やしたい

しかし、どうすれば「課題の発生を予め予測し、防止」できるのかがわからなかった。

講師が「運用あるある」で気付いたこと

対処3: 課題発生の「予防」を考えるようにした

「課題」に事後的に対応するのではなく、「予防」を考える。

事後的対応となる課題

課題の発生を予測できない

課題への対応を準備できない

課題の発生を予防できない

事前的対応できる課題

課題の発生を予測できる

課題への対応を準備できる

課題の発生を予防できる

Step3. 「運用」そのものの捉え方から考えなおす必要性に気付いた

「実装」を取り巻く環境やゲームルールが変わると、それが即課題の発生に繋がる事に気付いた。

「運用」を取り巻く環境やゲームルールも「運用」の設計で考慮する必要性に気付いた。

やっていることの価値やその変化の見通しが無いと、将来を見据えること(予防)はできない

講師が「運用あるある」で気付いたこと

参考: 運用改善も自動化も「銀の弾丸」ではないことに気付く

- ・ 「**運用改善**」しても「**自動化**」しても、先が見えてこない
 - ・ たしかに良くなっているが、根本解決した感じがしない。
- ・ **改善すればするほど、より大きな課題に直面して、ゴールが遠くなる**
 - ・ 「**課題解決**」という蜃気楼を追う気分。

「改善」は現在の延長線での「課題解決」だと気付いた

本当の意味での「根本解決」には、**破壊的な変更**や**非連続的な変革**が必要

「自動化」は単なるコード化だと気付いた

「人がやっていたことが機械に置き変わる」以上の効果はあまり期待できない。

「課題」を、事後的に点で潰していただけた。

「運用改善や自動化だけ」では、「課題」を面でまとめて潰すことや、**根本解決・予防はできない。**

気付き: 「運用改善」しても「自動化」しても先が見えない理由

1. 本当の意味での「設計」をしていなかった

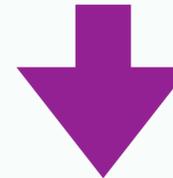
設計不存在

2. 「設計」の対象の捉え方が狭すぎた

視野狭窄

3. 「運用」の捉え方が適切ではなかった

存在意義曖昧



「運用の羅針盤」が存在しない

「運用改善」しても「自動化」しても次々と課題に追われる

講師が「運用あるある」で気付いたこと

みなさんの運用に「運用の羅針盤」はありますか？

「運用の羅針盤」のイメージ (例)

「課題」に対して面に対応をするための拠り所

「課題」を根本的に解決するための拠り所

「課題」を発生させずに、予防的に対応するための拠り所

正しい「運用改善」や「自動化」のための針路を示すもの

講師が「運用あるある」で気付いたこと

OpsLearn

運用設計ラーニング

OpsLearn

OpsLearn

運用設計ラーニング

当日の本編に続きます

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング

OpsLearn

運用設計ラーニング