

新技術を活用した システム開発・運用手法の研究

技術本部 システム研究開発センター

若林 耕平

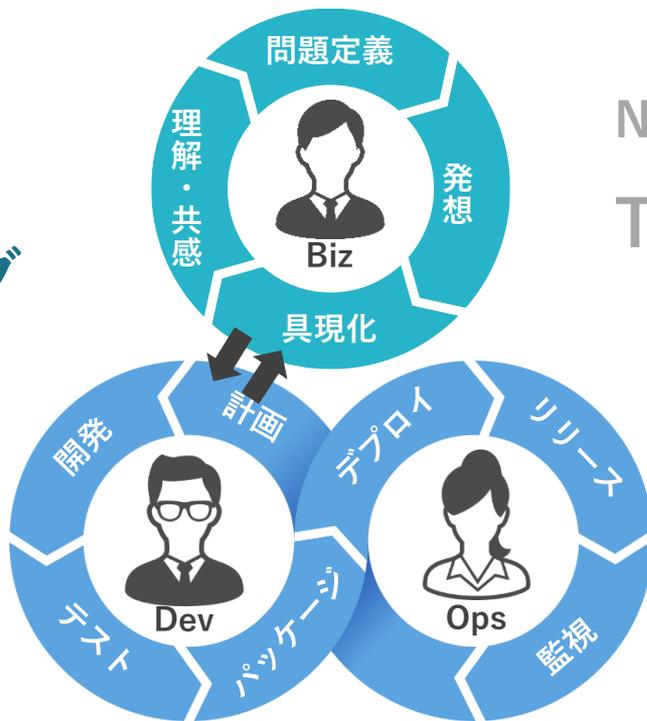
DX&イノベーションセンター

伊藤 宏樹

本セッションの位置付け

システムのアジリティと
信頼性を両立する

マイクロサービス・
クラウドネイティブ



NSSOL開発DXの実現

Tetralink

アジェンダ

1

アジリティを高める新技術の実用化

2

デザインパターンを活用したDXの推進

3

DXのさらなる推進に向けて

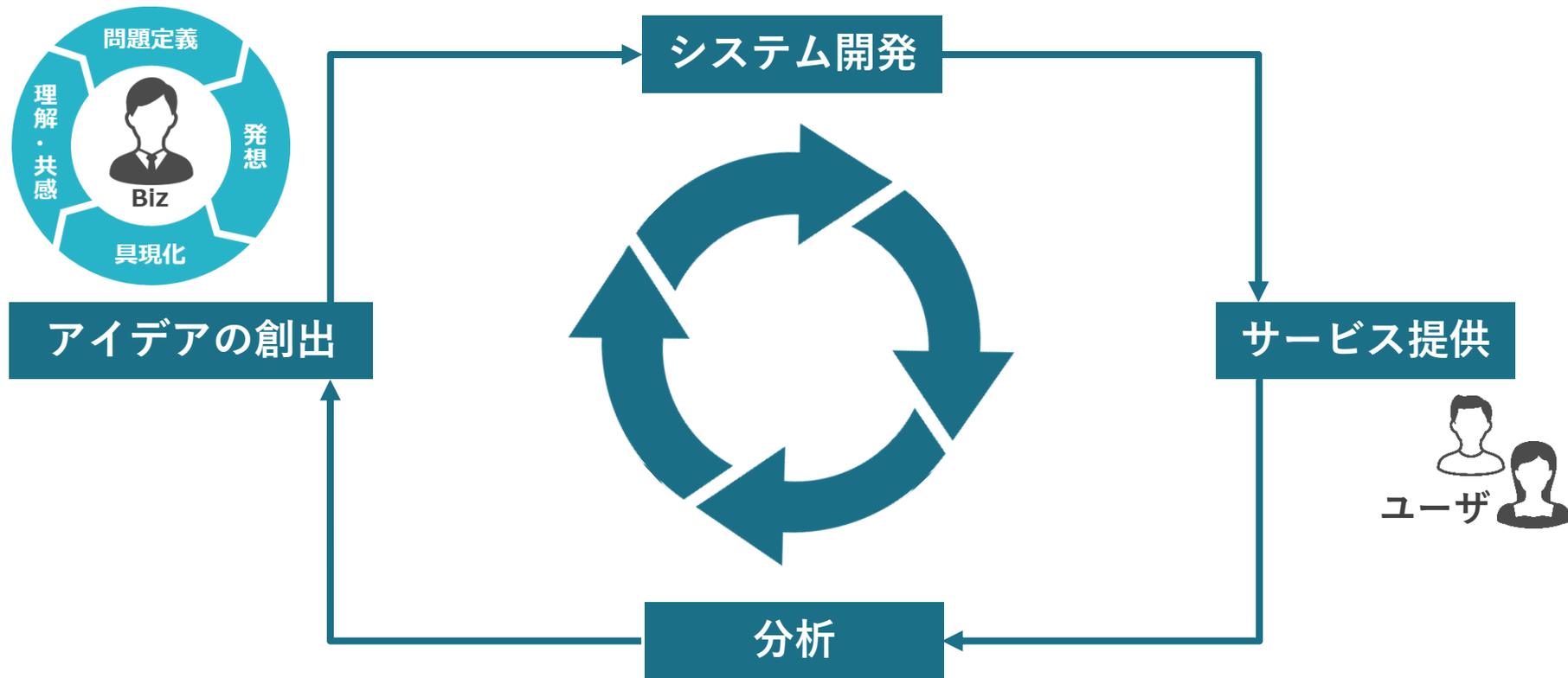
1

アジリティを高める新技術の実用化

システム研究開発センターのこれまでの取り組み

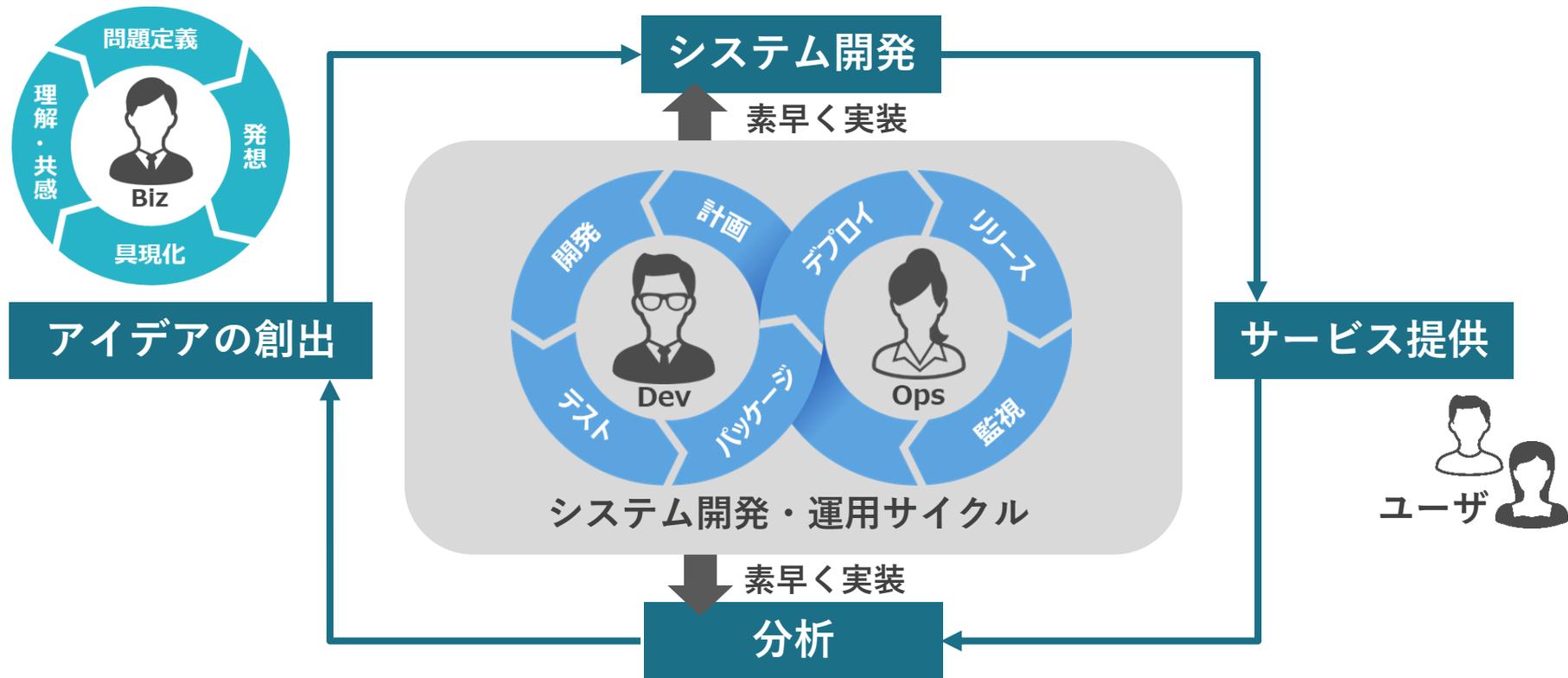
ビジネスが求めるシステムのアジリティ

ユーザ評価に基づくサービス改善やプラットフォームが求める早期のシェア拡大など



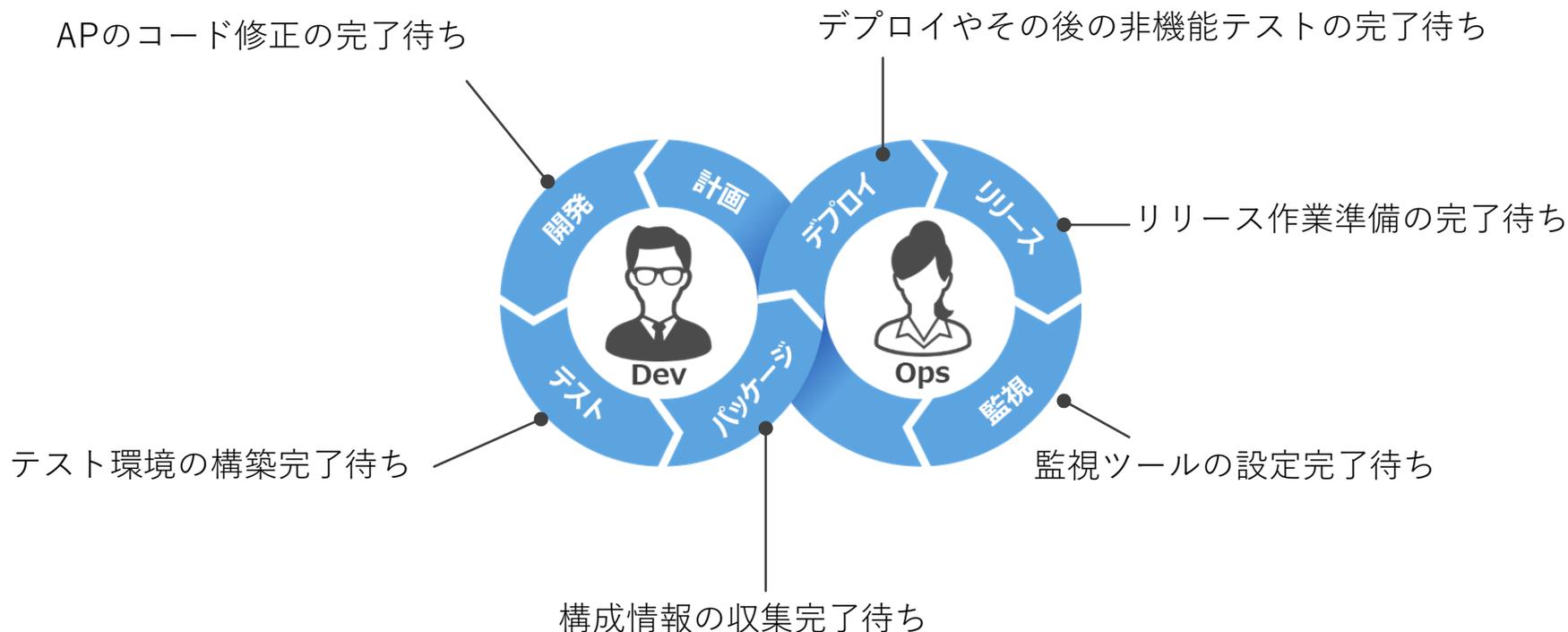
ビジネスが求めるシステムのアジリティ

ユーザ評価に基づくサービス改善やプラットフォームが求める早期のシェア拡大など



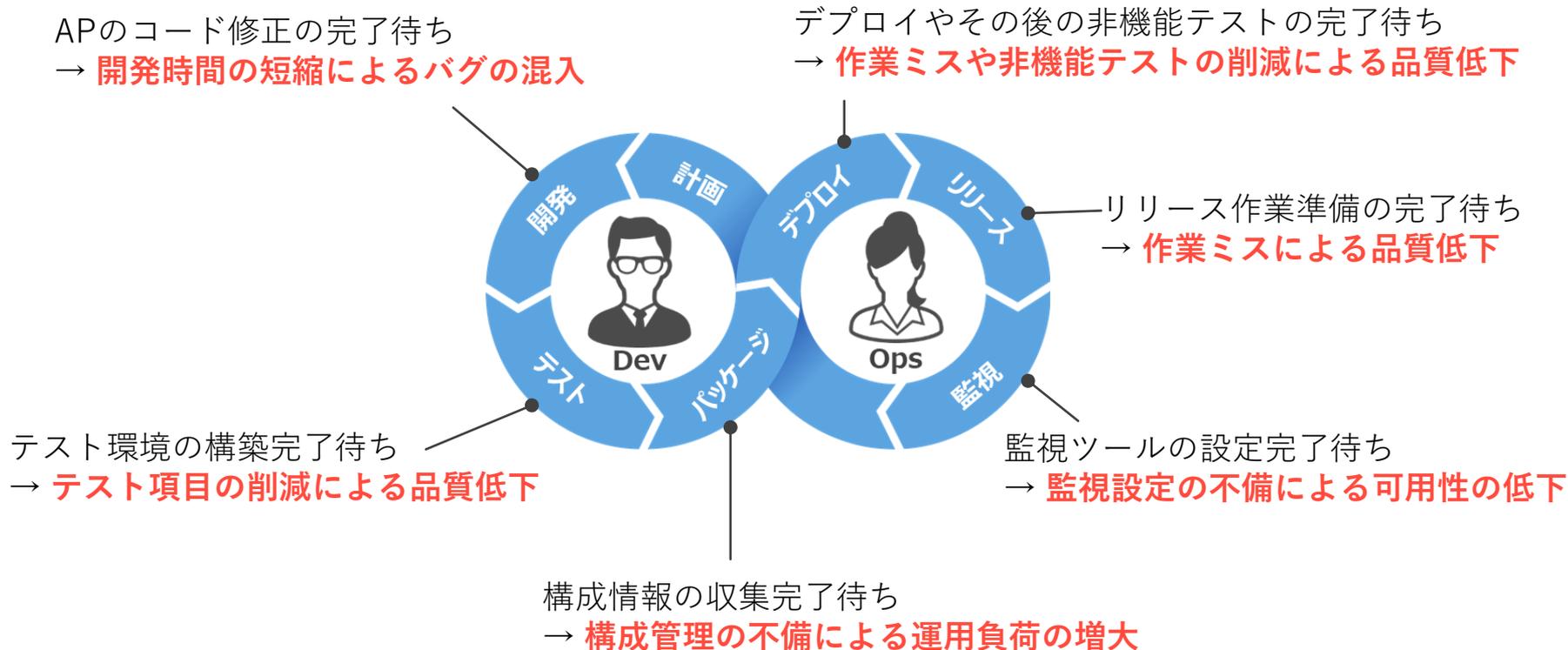
従来の手法の限界

手法を変えずにアジリティを高めようとすると、信頼性が犠牲になる



従来の手法の限界

手法を変えずにアジリティを高めようとする、信頼性が犠牲になる

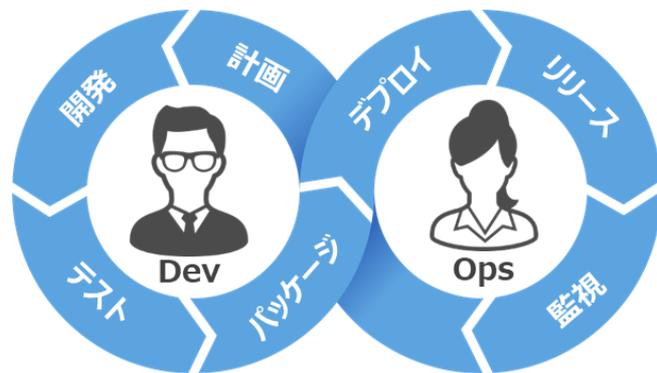


アジリティを高める新技術

2つの技術で信頼性を損なわずにアジリティを高める

マイクロサービス・アーキテクチャ

APを複数の小さなサービスに分割し、サービス単位での独立した開発を可能にする



クラウドネイティブ

コンテナやkubernetesを用いてシステム開発・運用業務を高度化・自動化する

アジリティを高める新技術

マイクロサービスとクラウドネイティブで従来の手法の限界を超える

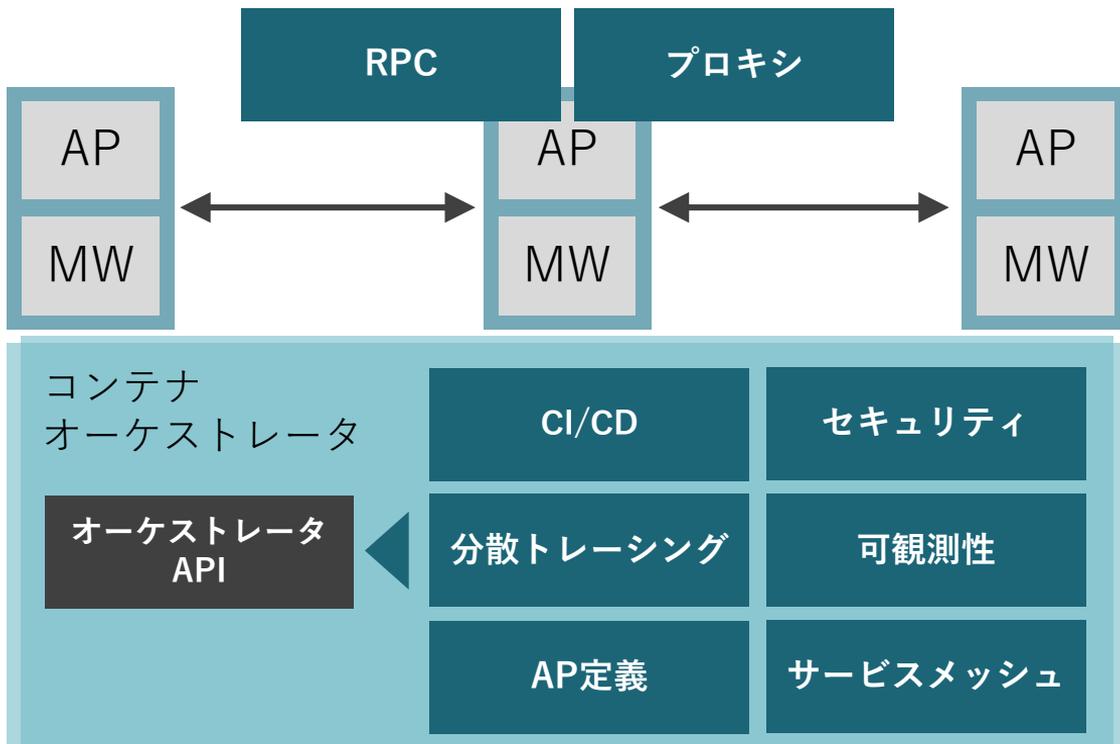
複数のマイクロサービスに分割し、



アジリティを高める新技術

マイクロサービスとクラウドネイティブで従来の手法の限界を超える

オーケストレータで管理する



コンテナ技術がオープンなコミュニティで急速に進化。

マイクロサービスの開発・運用を支える多数のツール・サービスを提供。

アジリティを高める新技術

マイクロサービスとクラウドネイティブで従来の手法の限界を超える

オーケストレータで管理する

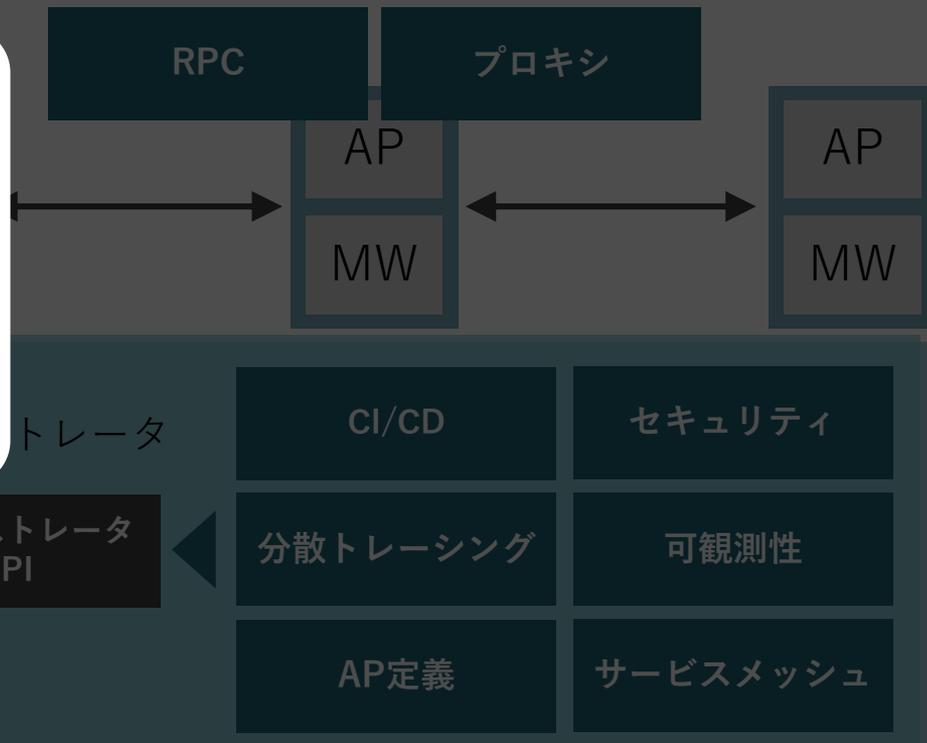
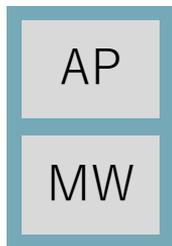
- ・開発時の検討対象を絞り込める
- ・サービス単位でデプロイできる



従来の複雑さやデプロイの問題に対処



Dev



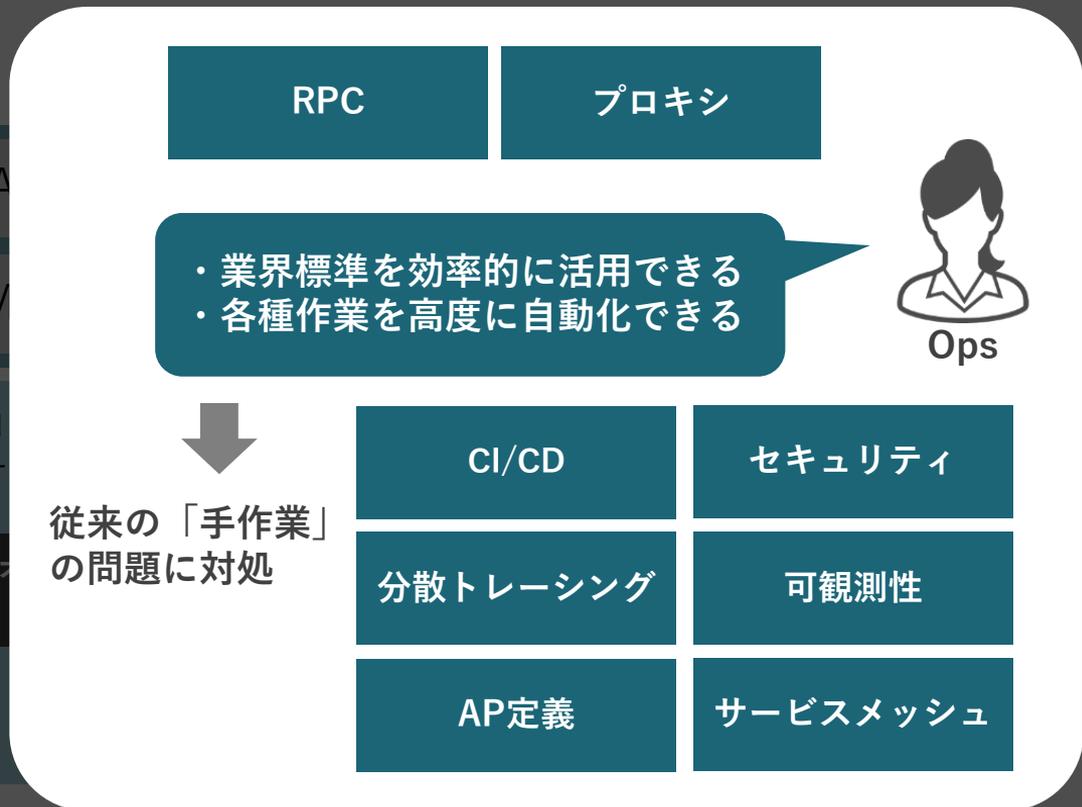
アジリティを高める新技術

マイクロサービスとクラウドネイティブで従来の手法の限界を超える

オーケストレータで管理する

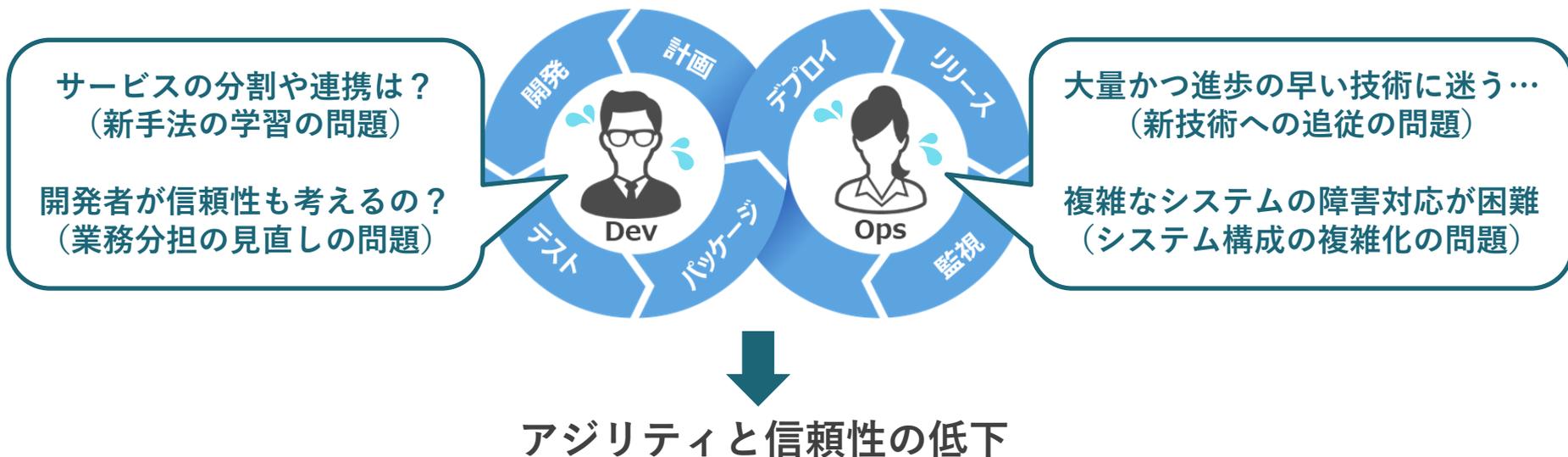
コンテナ技術がオープンなコミュニティで急速に進化。

マイクロサービスの開発・運用を支える多数のツール・サービスを提供。



新技術の難しさ

新技術に伴う様々な問題を解決しなければ、むしろ逆効果になる



その難しさを解消する方法

新技術を活用したシステム開発・運用手法をデザインパターンとして整理

Microserviceの設計・実装

サービス分割ガイド <ul style="list-style-type: none">マイクロサービスの考え方サービス分割の進め方サービス間通信の設計ベースアーキテクチャなど	API設計ガイド <ul style="list-style-type: none">REST API 設計ガイドAPI GW 活用ガイド認証認可設計ガイドなど
設計・開発支援 <ul style="list-style-type: none">設計の進め方技術者の育成成果物を活かしたアドバイスなど	実装テンプレート <ul style="list-style-type: none">AP開発テンプレートAPテストテンプレートシステムテストテンプレートなど

開発面単位での環境管理

環境構成のPR Dev → A環境構成 / B環境構成	PR承認 = 払い出し承認 承認した変更のみMerge A環境構成 / B環境構成 → Ops
Devへの作成完了通知 デプロイツール → Chat オークストレータ → A環境 / B環境	ツールによる環境作成 差異を検出 → 構成リポソトリ デプロイツール → オークストレータ → 適用

デプロイの信頼性向上

必須設定のチェック 設定確認ツール → オークストレータ オーバーシフupは正しいか? / タグは正しいか? → A環境構成 / B環境構成	差分検知をCIで自動化 構成リポソトリ ↔ オークストレータ
費用対効果の向上 Dev → Push → Appリポソトリ → CI/CD → コンテナレジストリ → デプロイツール → オークストレータ → Push → Dev 信頼性が高く運用負荷も下がる!!	デプロイ機構の監視 監視ツール ↔ デプロイプロセス ↔ オークストレータ

分散トレーシングによる可観測性の向上

サービス障害の検知 SRE → ダッシュボード → 障害発生	高可観測性ツールの活用 トレース・メトリック ログ 次世代型 APM ツール / 次世代型 ログ管理 ツール
サービス障害からの復旧 SRE → ダッシュボード → 復旧完了	原因特定と修正適用 SRE → Push → Appリポソトリ / 環境構成リポソトリ

Shift-Left型のコンテナセキュリティ

ソフトウェアの追加 Dockerfileで Rubyを追加 → Appリポソトリ	CIでの脆弱性スキャン CIプロセス → コンテナイメージ (Ruby) → コンテナセキュリティツール → CVEに基づき Rubyの脆弱性を検証
CIでのポリシー検査 CIプロセス → 環境構成ファイル → ポリシー管理ツール → ポリシーの適合性を検証	環境構成定義の更新 Dev → 環境構成定義を更新 → 環境構成リポソトリ

APレイヤも含めた可用性の担保

Node障害の発生 オークストレータ → 監視用node / node / node / 100% → 障害発生	Circuit Breakerの発動 User → コンテナ → コンテナ → コンテナ → 障害発生時Circuit Breakerでコンテナを切り離す
その間に自動修復 オークストレータ → auto healing → 監視用node / node / node / node → 経路復旧	その間サービスは継続 User → 200 OK → サービス → 一部の機能は使えないがシステム全体の停止を回避

その難しさを解消する方法

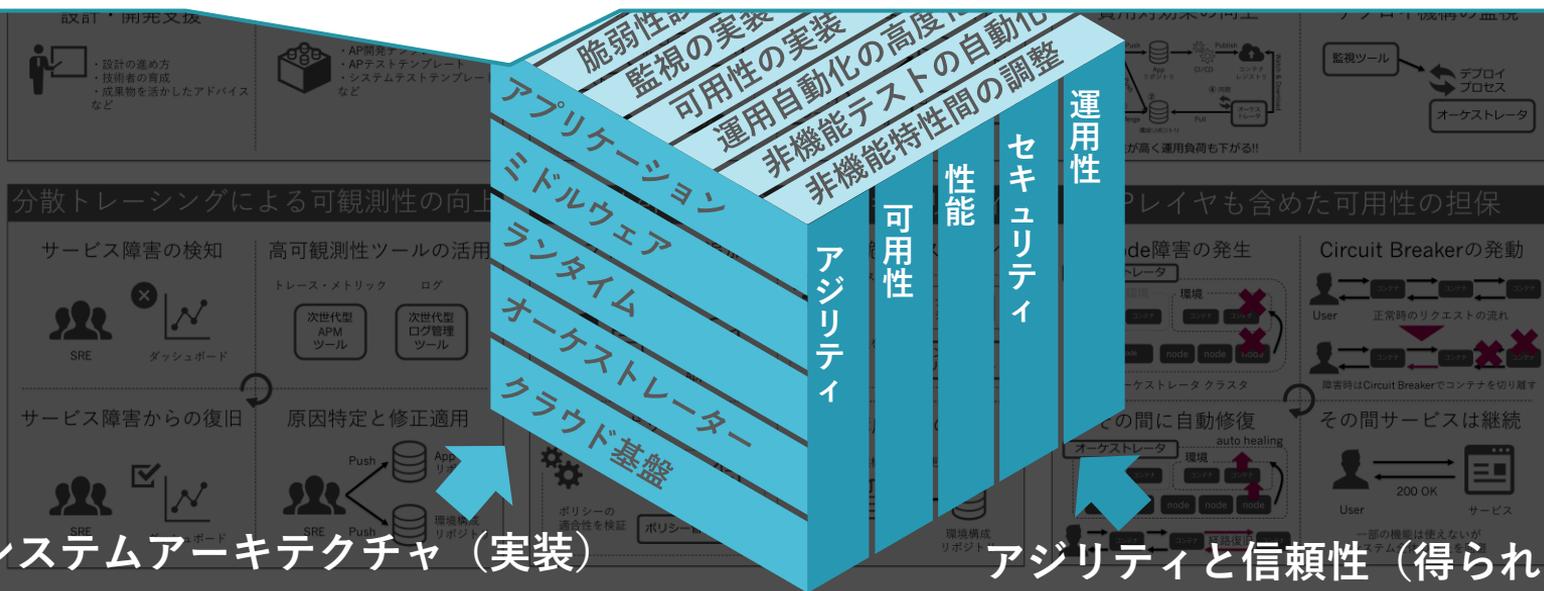
新技術を活用したシステム開発・運用手法をデザインパターンとして整理

Microserviceの設計・実装

開発・運用プロセス (実装)

デプロイの信頼性向上

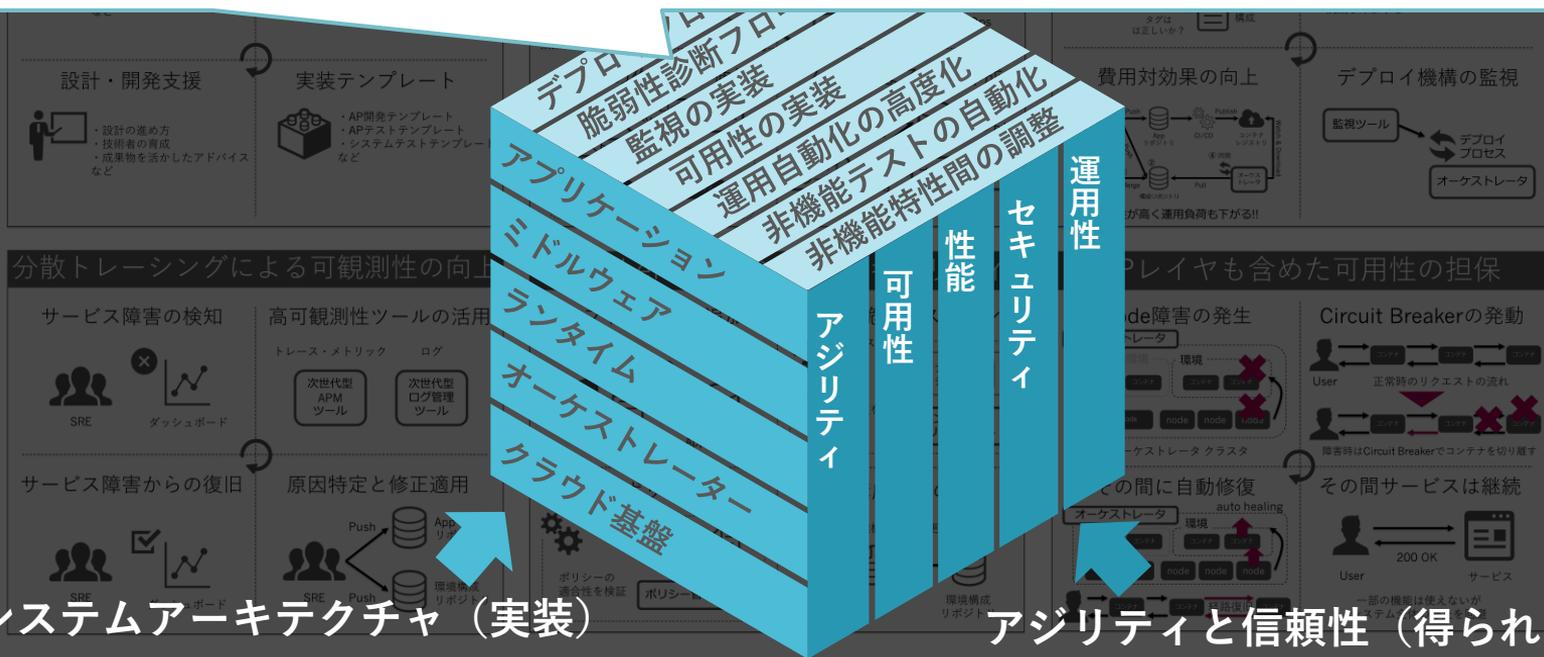
新技術で「構成の複雑化」に対処するにはAPとインフラを融合したシステム設計が必要
→ NSSOLはワンストップの強みを活かして、その手法をデザインパターンにまとめた



その難しさを解消する方法

新技術を活用したシステム開発・運用手法をデザインパターンとして整理

新技術で「業務分担」の問題に対処するには開発と運用を融合した新しい業務プロセスが必要
→ NSSOLはワンストップの強みを活かして、その手法をデザインパターンにまとめた

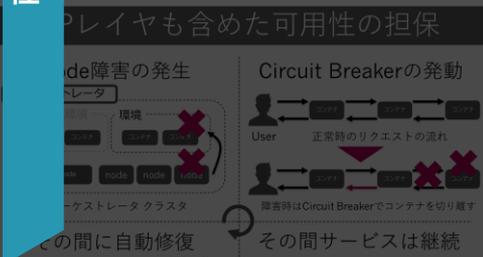
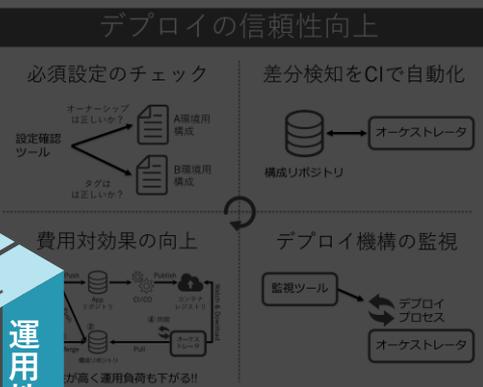


その難しさを解消する方法

新技術を活用したシステム開発・運用手法をデザインパターンとして整理

新技術の活用に伴う「技術選定」の問題に対処するにはツールの目利きや検証ノウハウが必要

開発・運用プロセス (実装)



システムアーキテクチャ (実装)

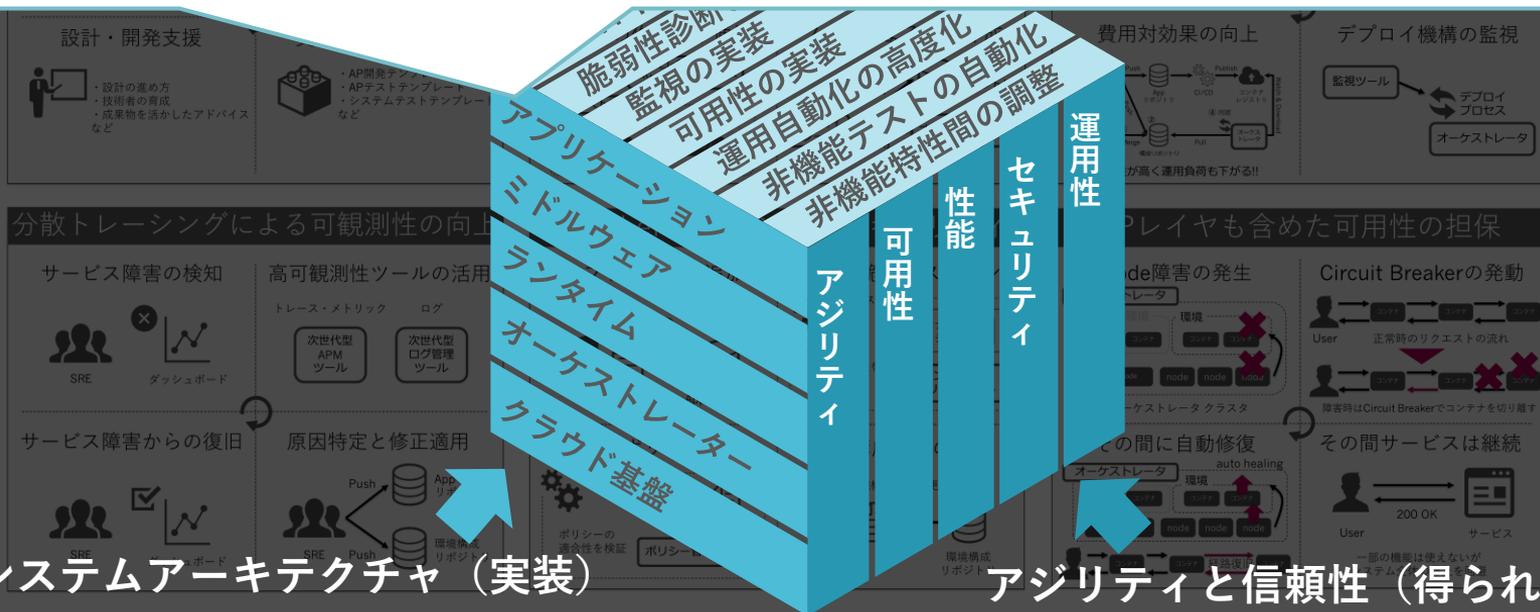
→ NSSOLは大規模システムの開発・運用経験で把握したアジリティや信頼性の具体的な低下要因の解消を図れる

その難しさを解消する方法

新技術を活用したシステム開発・運用手法をデザインパターンとして整理

開発・運用プロセス (実装)

新技術の活用に伴う「学習」の問題に対処するには実践的なエンジニアリング手法の整理が必要
→ NSSOLは各業界での開発経験で得た業務知識を活かしてマイクロサービスの適用を実践中



その難しさを解消する方法

デザインパターンでアジリティと信頼性に優れたシステムを効率的に実現する

ベース

デザインパターン

問題の解決

DXへの貢献

コミュニティの
コンテナ技術

NSSOLの強み
・ワンストップ体制
・各業界での大規模開発

APとインフラを融合する
新しいアーキテクチャ

開発と運用を融合する
新しい業務プロセス

期待する効果を得られる
ツールとアーキテクチャ

マイクロサービスの
実践手法

構成の複雑化

業務分担

技術選定

学習

解決

解決

解決

解決

10案件に適用

アジリティと
信頼性の向上

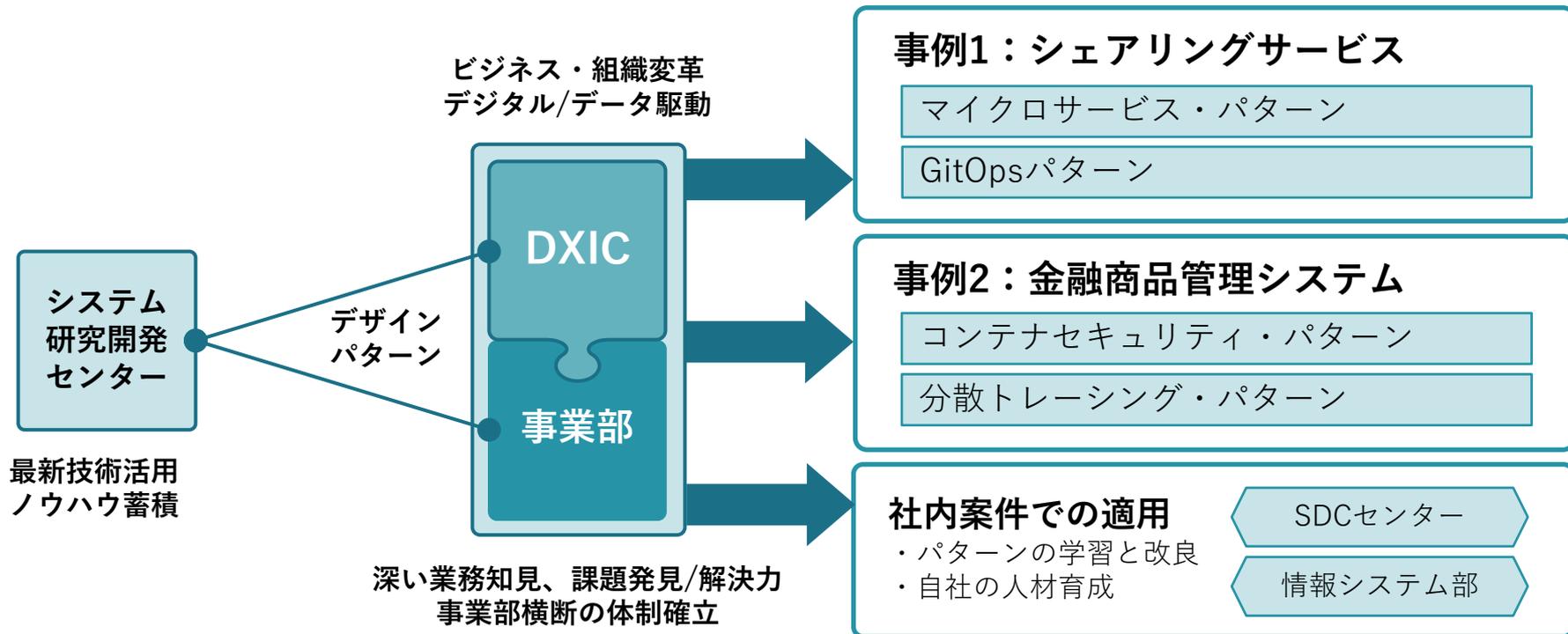
2

デザインパターンを活用したDXの推進

DXICのDX案件事例

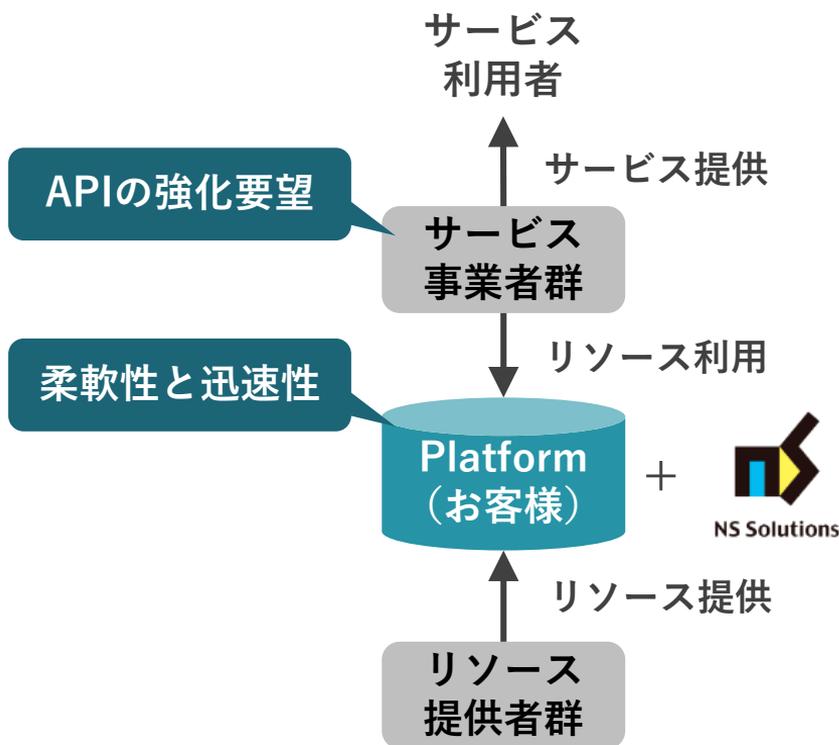
新技術を活用したDXの推進

デザインパターンを習得したDXICが事業部横断の体制を技術でリード



事例1：シェアリングサービス

お客様のビジネスが求めるシステムの柔軟性と迅速性を新技術で実現



事業の方針をお客様から共有して頂く過程で「今まで以上の柔軟性や迅速性」が必要であり「社会インフラとしての信頼性」も担保すべきことをNSSOLは強く認識。

新技術の可能性とリスクを評価した上で以下のパターンをお客様に提案。

<採用したデザインパターンの例>

マイクロサービス・パターン

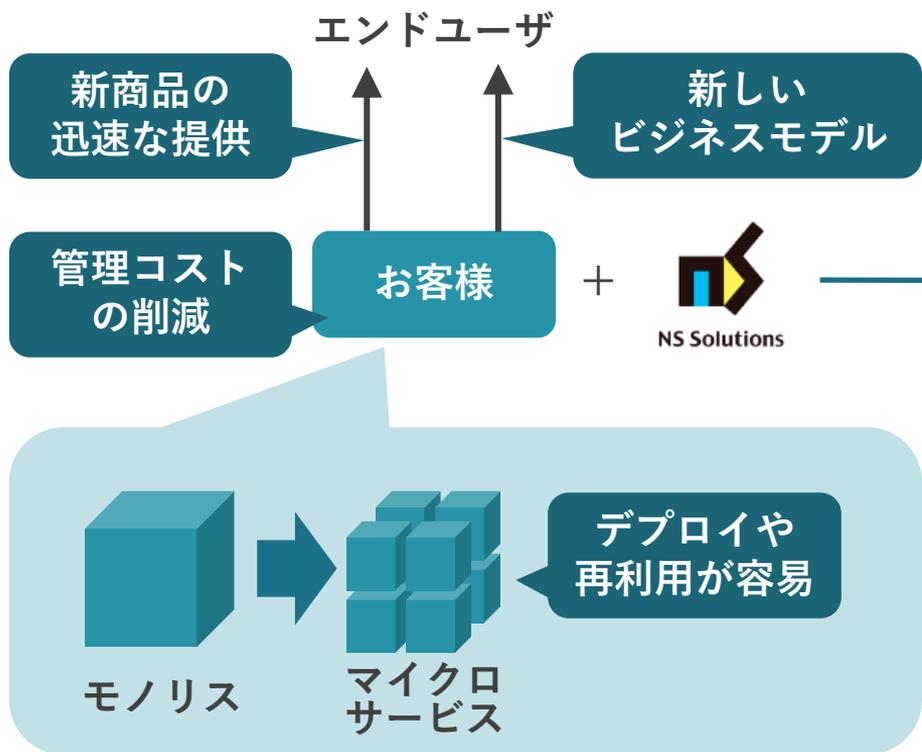
指針やテンプレートによる設計の効率化と品質向上

GitOpsパターン

デプロイのセルフサービス化によるアジリティ向上

事例2：金融商品管理システム

アジリティに加えて、システムの可用性やセキュリティも追求



お客様の業務の理解を進める過程で、アジリティだけでなく、業務の継続性を高めるシステムの信頼性も重要と認識。

マイクロサービスの信頼性を高める以下のパターンを提案。

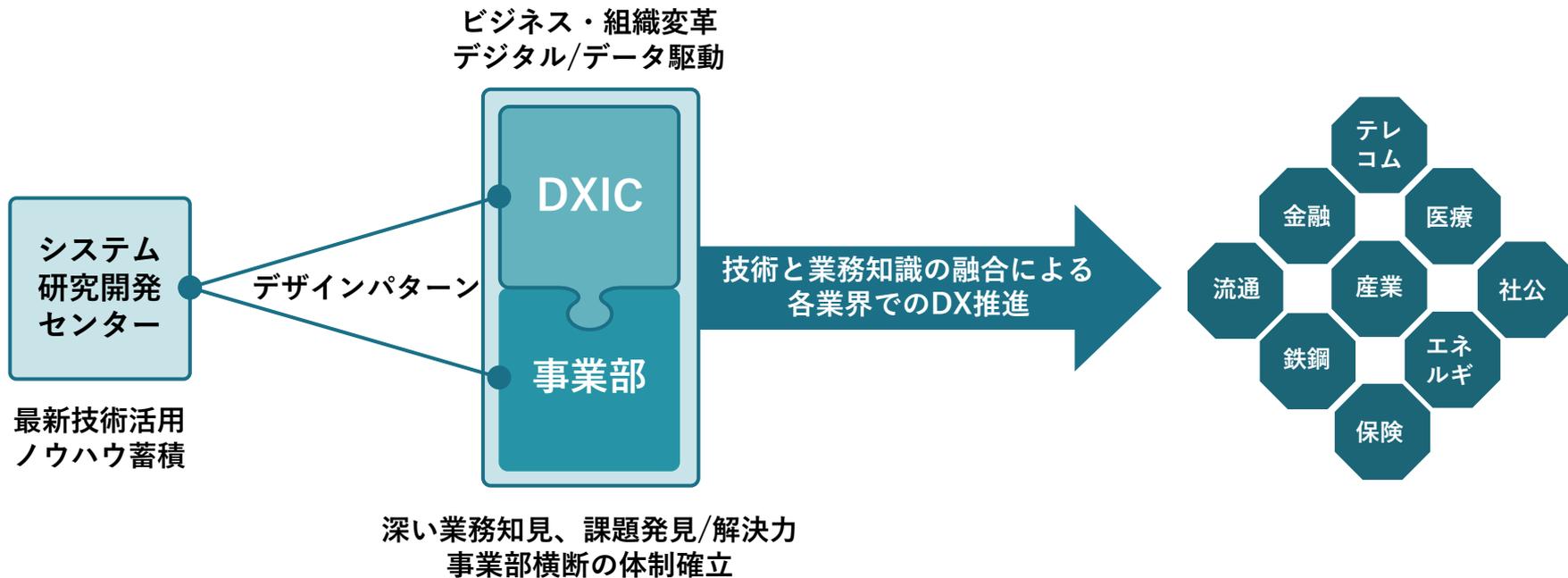
<採用したデザインパターンの例>

コンテナセキュリティ・パターン
脆弱性診断のDev向けセルフサービス化

分散トレーシング・パターン
システム状態の可視化による障害対応力の向上

新技術を活用したDXの推進

デザインパターンを習得したDXICが事業部横断の体制を技術でリード



現場からのニーズ

DXに不可欠な新技術をデザインパターンを通して継続的に担保する

今後の動き

コミュニティによる
継続的な技術革新

お客様の
DXの取り組み強化



現場のニーズ

デザインパターンの
継続的な開発

デザインパターンを
活用できる人材の育成

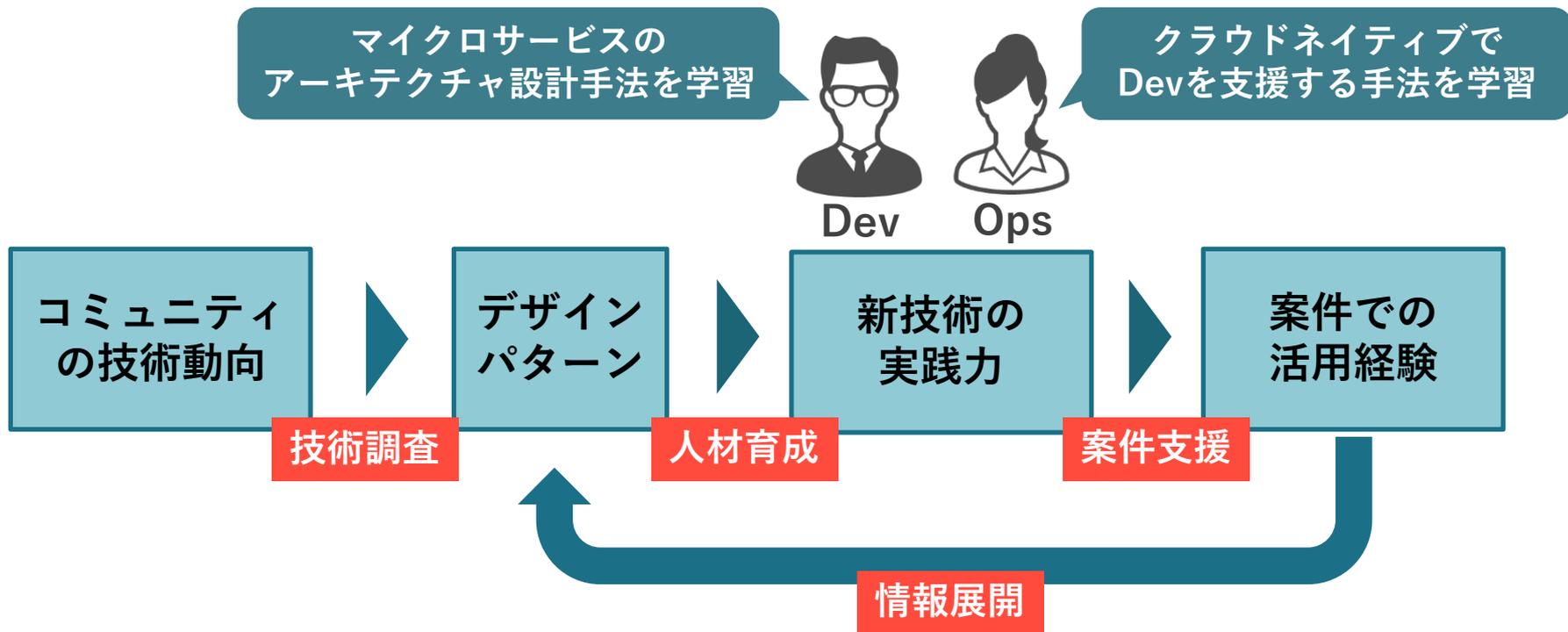
3

DXのさらなる推進に向けて

システム研究開発センターの今後の活動方針

現場のニーズに継続的に対応

デザインパターンを媒体にした技術力の強化を推進する



商標について

X Integrator、hifive、Geminant、Geminant（ロゴ）、安全見守りくん、IoX、IoXプラットフォーム、匿丸、匿丸（ロゴ）、OptBrain、OptBrain（ロゴ）、Tetralink、Tetralink（ロゴ）、日鉄ソリューションズ、NSSOL、NS(ロゴ)は日鉄ソリューションズ株式会社の登録商標です。

その他本文記載の会社名及び製品名は、それぞれ各社の商標又は登録商標です。