クラウド時代のエンタープライズDevOps

http://www.Ask3S.com

Strategic Staff Services

2014/4/16

株式会社　戦略スタッフサービス
戸田　孝一郎　CSM
(社)TPS検定協会　理事

ScrumAlliance
transforming the world of work

社団法人TPS検定協会

はじめに

DevOps ＝ Development ＋ Operation

開発と運用の統合、連携？？？
開発と運用のコラボレーション＆コミュニケーション

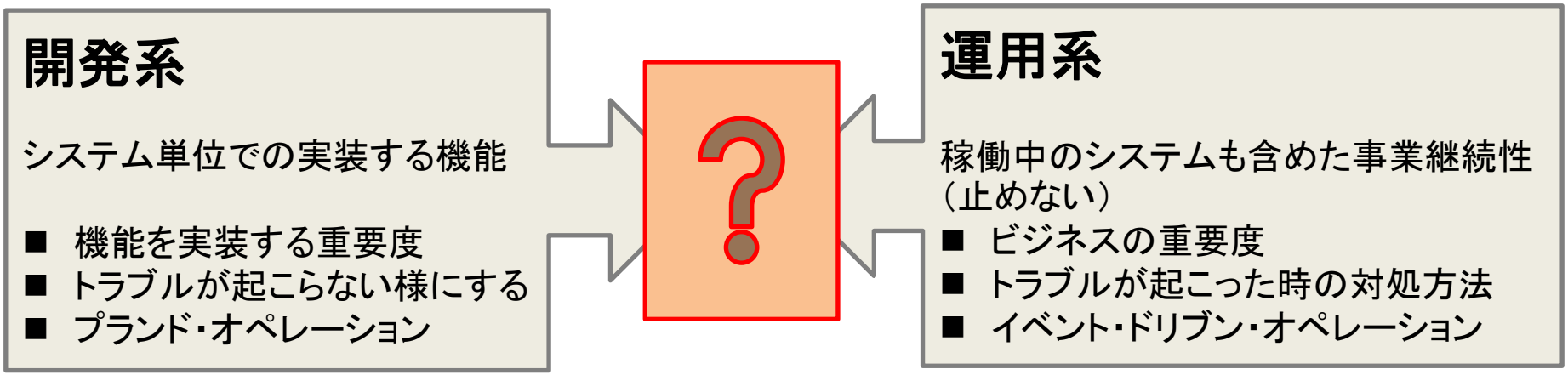
企業におけるITの価値の本質に立ち返る活動
ITの価値の本質　＝　ビジネスをタイムリーに支援する。
ビジネス・スピードを牽引する。
ビジネス領域を拡大させる。
見えなかったモノを見える様にする。

JITでのITサービスの提供

課題が見えてきた

会話（コミュニケーション）が成り立たない。

原因は、
視点の違い。優先順位（価値観）が異なる。



ITIL® : 内部統制の視点からは、開発と運用の役割と権限の分離を求められる。

ユーザーにITサービスをタイムリーに提供する

ITサービスの提供とは、（構成要素）
ビジネス・プロセス　＋　ソフトウエア　＋　ITインフラストラクチャー

全ての基準は、ビジネス価値

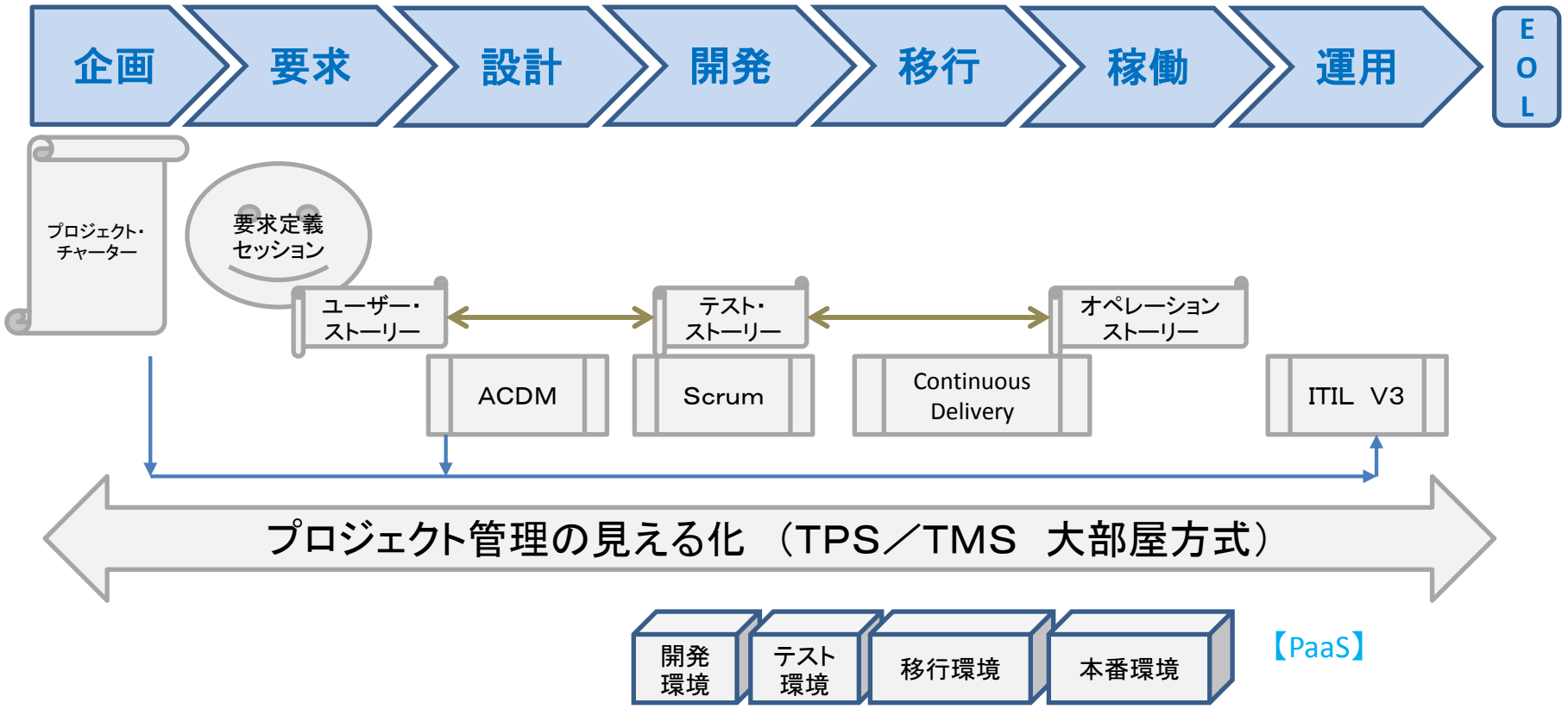
提供するITサービスのビジネスへの貢献度
ビジネスへの貢献度が失われれば、EOL（End of Life）
貢献度が下がれば、更新（Update）

ユーザーにITサービスをタイムリーに提供する手法

DevOps

企画からEOL迄、一貫したプロセスの見える化と管理
【トヨタ生産方式（トータルTPS/TMS）で言われる大部屋方式】
ITサービスを提供する関連情報の一元化（共有化）も必要
特にBuild, Test, Deployの手順化＆自働化の検討も必須

Enterprise DevOps



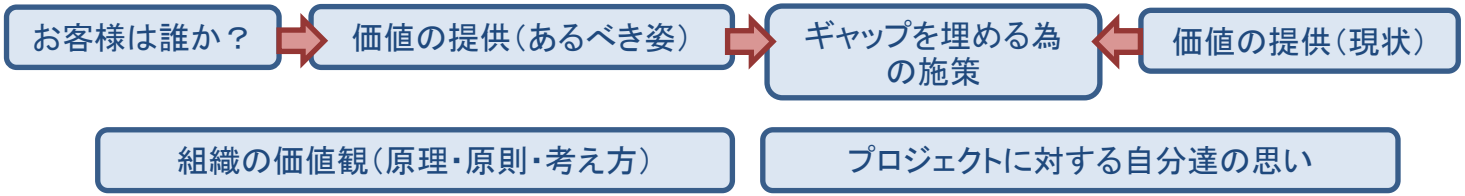
プロジェクト企画

Project Charterの作成 （LOB ＋ CIO）
事業部門（ユーザー）とIT間での合意

【内容】
ビジネスのビジョン、戦略（掛けられるコスト、期間）
プロジェクトの目的（目標、期待効果）
組織体制（責任者の権限と責任）
役割の定義（責任）
EOLの要件（条件）

以上を明示して、プロジェクト組織（チーム）を編成する。
そのうえでメンバー全員がこれら（Project Charter）を理解し、共有する。

プロジェクトの方針展開
上位方針を受けて下位のチームと擦り合わせを行い具体的な行動を起こせるようにプロジェクトとしての目標やあるべき姿、基本的価値観の共有を図る



システム要求の定義

要求定義セッション

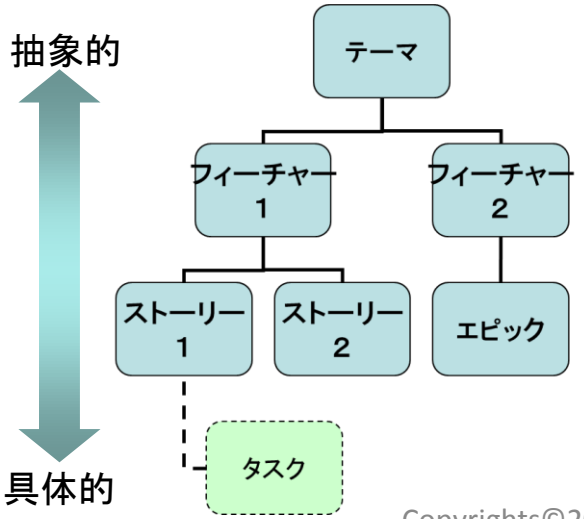
参加者：	プロダクト・オーナーとユーザー代表
	スクラム・マスターと開発チーム（リーダーとメンバー数名）
	サービス・アーキテクト（運用のSE）
オブザーバー：	プロジェクトの管理者
スポンサー：	事業部長（LOB）、CIO

【内容】

システム全体像（システムの目的）の合意、イメージの擦り合わせ

初期のプロダクト・バックログの確定（内容＆優先順位）

- ◆ユーザーストーリー（要求をビジネス価値で表現）、
- ◆テスト・ストーリー（受け入れ基準）、
- ◆オペレーション・ストーリー（トラブル時の運用の対処）の確認

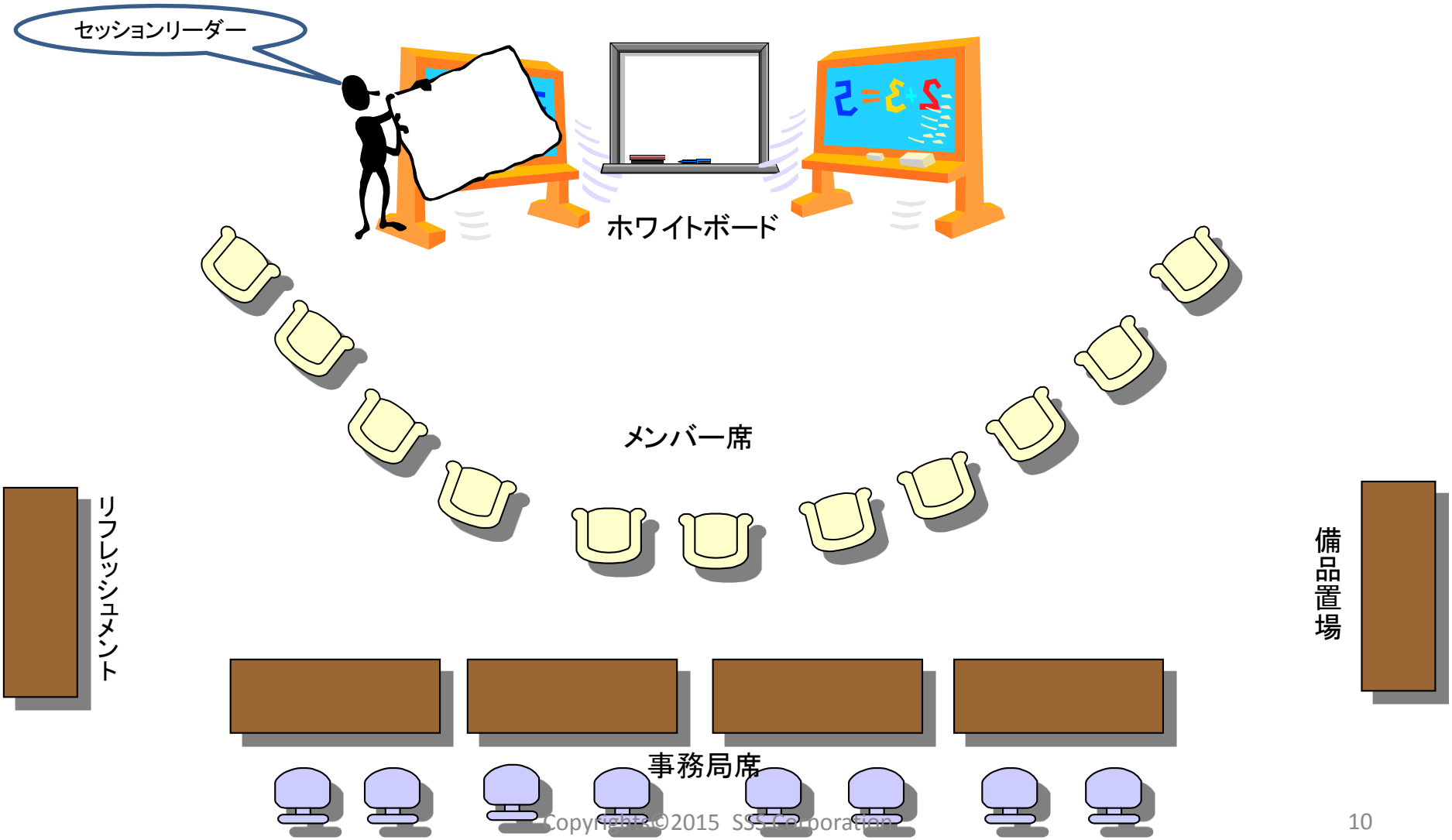


User Story: User Stories Applied by Mike Cohn

要求定義セッションの流れ

- 事前準備
  - 対象とするシステムの範囲の明確化
  - 参加メンバーの決定
  - 事務局の構成決定
  - 対象となるシステムの制作目的（ビジネス的意義、価値）やビジョン
  - システムに要望するユーザーとしての要求事項の整理（優先順位付けされた要求の下案）
  - 関連する情報（データ）の収集＆整理
  - セッションの日程、会場の手配
  - 参加メンバーへの招待状（セッション主旨とルールの説明）
  - セッション機材の手配
- セッション
  - セッションルールの説明（再確認）
  - アイスブレーク（参加者全員の自己紹介）
  - セッション実施（1テーマ毎）
  - 決定事項確認（次のアクションの確約）
- 事後作業
  - 議事録の作成
  - 関連資料（アウトプット）の整理＆作成
  - 参加者への配布

セッション・ルーム配置図



セッション・ルール

- 第三者のセッション・リーダーによる100%コントロール・ディスカッション形式
  - 発言はセッションリーダーの了解を得て行う
  - 一時点では一人の発言者で、常に発言者とセッションリーダー間のダイアログの形式になる。
  - 発言者の話が終わるまで待つ。対案がある時は挙手をしてセッションリーダーに意思表示を行う。
- 参加者は全員対等
- 一時点では一つのテーマ（話題）の議論に集中する。
  - 議論の過程で派生する話題（テーマ）はセッションリーダーがメモに書き留め、別の時間帯で議論を行う。
- 決定はメンバー全員の合意による
  - 合意するまで議論を尽くす。
  - 多数決による決定は行わない。
- 発言者は議論している内容について、賛成、反対を明確に意思表示する。またその理由の説明責任を有する。
- 発言は、簡潔明瞭に適語表現で行う。
- 出来るだけデータに基づき議論を展開する。
  - 感情や気分による発言は避ける。（やりたくない。なんとなく気が向かない。と言った理由）
  - 不足しているデータは収集する。
- 参加メンバーはセッション中は、頭脳をフル回転させ、100%議論に集中する。事務局がデータの整理や、補足情報の提示等、参加メンバーが議論に集中できる環境を支える。
  - セッション・ルームには、メンバーは筆記用具の持ち込みを禁止する。（メンバーの席には机を配置しない）
  - 議論のメモ等はセッションリーダーが前のホワイトボード等に簡潔に適語で書き留める。
  - 事務局が議事録を作成する。

ユーザーストーリーとは？

ユーザーストーリーは、ユーザーやシステム、ソフトウエアの利用者に価値をもたらす機能的なモノを記述します。

- ユーザーストーリーとは、ビジネス分析の表現手法
- 誰でも記述できる（専門知識不要）
- 簡潔・明瞭な表現

表現は、

As a role（役割、利用者）
I/we can ---（機能：何々ができる） ＋ which I need---（非機能要件：品質等）
To ---（ビジネス価値、効果）

予約受付係り＜**役割**＞として、

私は年間最繁忙期でも最低12件の旅行予約を60分以内に処理＜**機能**＞できる。

それは予約電話の不受信を最小限に収める＜**価値**＞ためだ。

エピック：
まだ明確に表現できないモノや緊急度の低いモノ、Nice to Haveな事柄は、大きな塊（エピック）として置いて置く。必要に応じてユーザーストーリーとして詳細化する。

ユーザーストーリーの簡単なルール　INVEST

Independent（独立性）

ユーザーストーリーは各々が独立している事

Negotiable（交渉可能）

ユーザーストーリーを使って、ユーザーと開発者が会話できその結果をまとめる。

Value to Users/Customers（価値を提供）

一つ一つのユーザーストーリーがユーザーにとって価値を提供できる。

Estimatable（見積可能）

開発者が作業を見積もれる事。

Small（小さい、簡潔）

簡潔な表現で、できるだけ小さな単位にする。

Testable（検証可能）

ユーザーがユーザーストーリーを基に受入テストができる事。

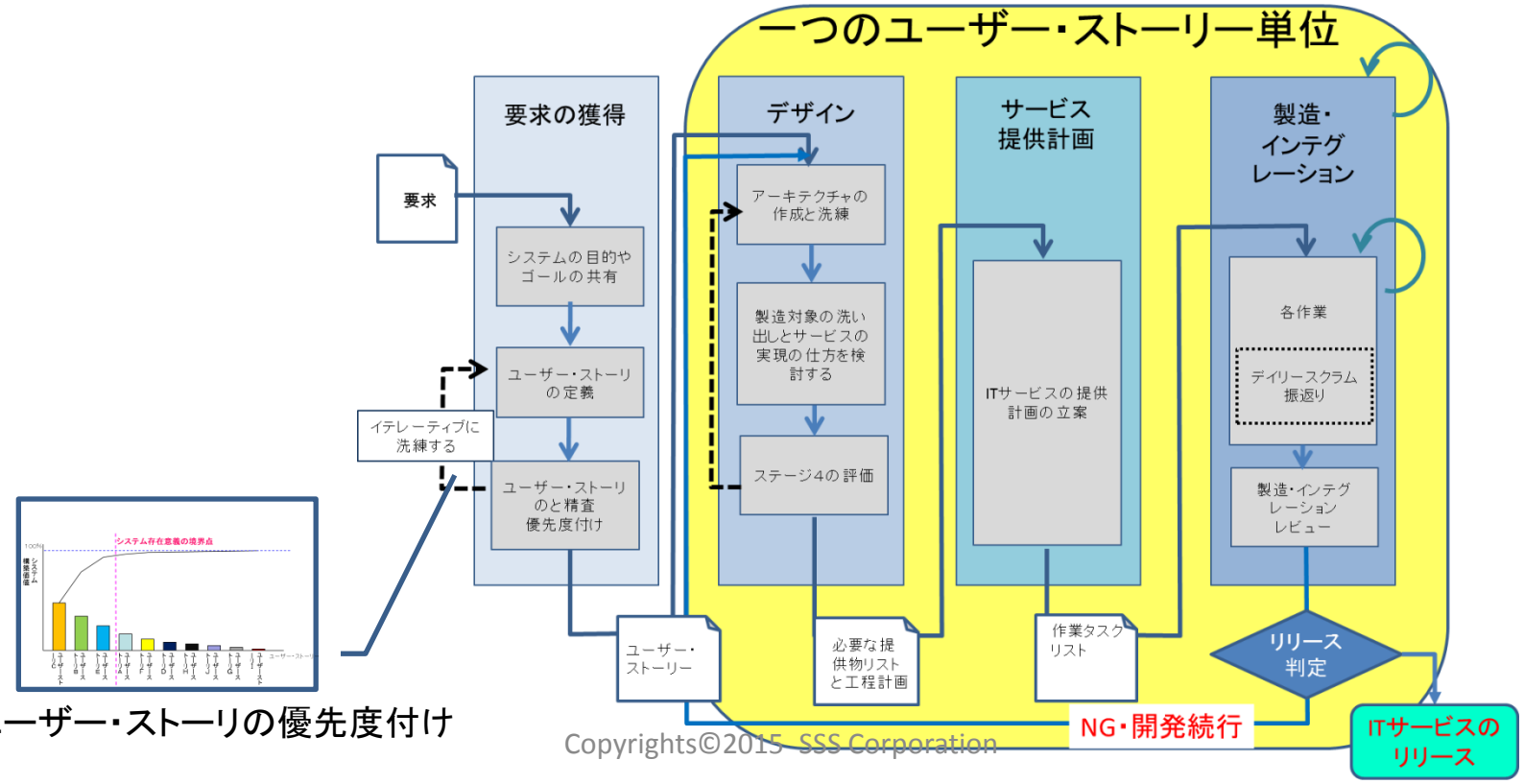
システムの設計

アーキテクチャ･ドライバーの明確化

- ◆RF　高次のシステム要求（機能）
- ◆RQ　品質特性要求（非機能要件）
- ◆BC　ビジネス制約（コスト、期間も含まれる）
- ◆TC　技術制約（技術＋開発環境、テスト環境、移行環境、本番環境、SACの選択）

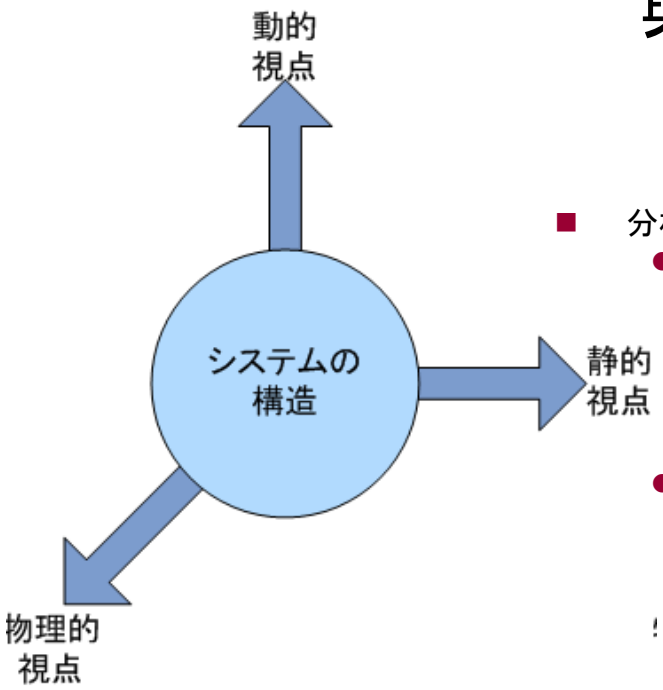
ACDM（アーキテクチャ中心設計手法）によるイタラティブなアプローチによるデザインの洗練

ACDM：The Architecture Centric Design Method by Anthony J. Lattanze



ユーザー･ストーリの優先度付け

経営/管理者層・開発・運用/オペレーションで興味分野が異なる

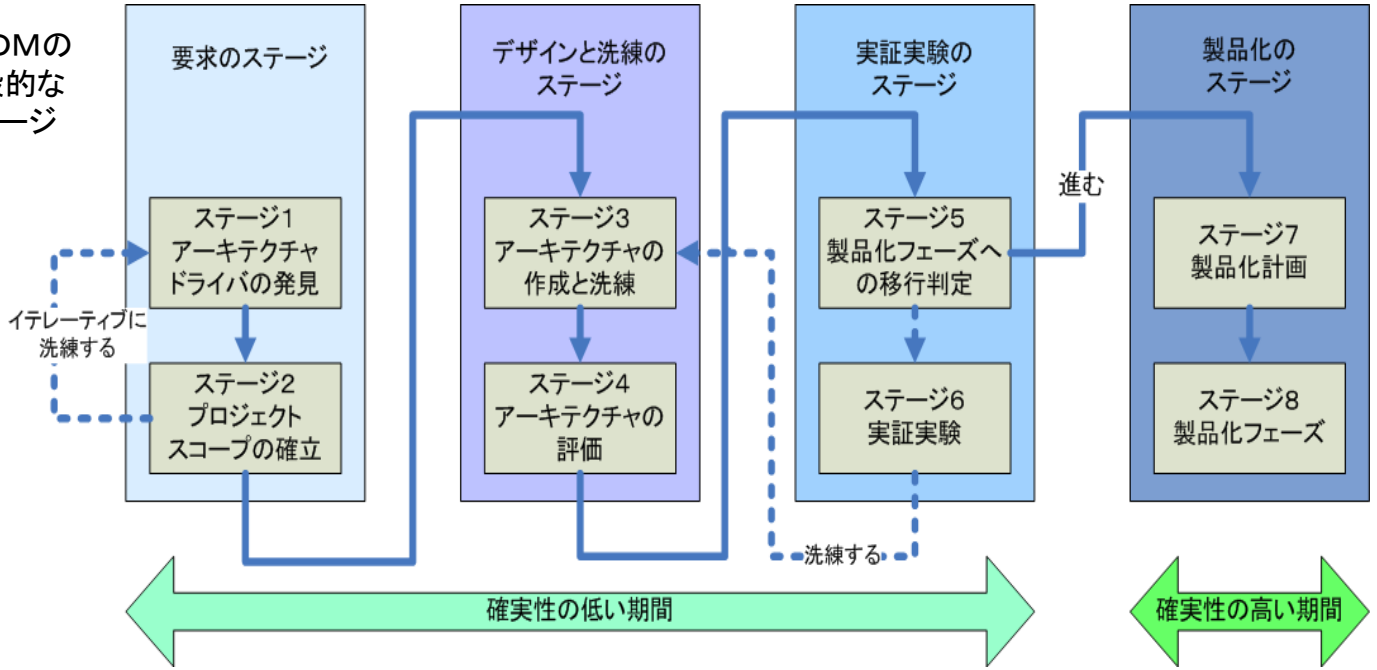


- 分析・設計における基本的な視点
  - **動的視点（経営層・ユーザー層）**
    - 実行時の構造を表し、実行時の運用に関係する関心事を分析する
      - 並行性、リソースの活用、負荷、パフォーマンス、可用性、異常検出とリカバリなど
    - 構造：プロセス、スレッド、オブジェクト、データレポジトリなど
    - 関連：データフロー、制御フロー、イベント、コール・アンド・リターンなど
  - **静的視点（開発者）**
    - クラスや関数といった細かな構造を束ね、アーキテクチャ要素を表現する大きめの抽象物として考える
    - 実装指向の事柄やコード指向の変更のコストに関する理由を説明する
      - 変更容易性、拡張性、スケーラビリティ、再利用性、移植性など
    - 構造：モジュール、クラス、データベーステーブル、ファイル、データ構造、レイヤーなど
    - 関連：使用、継承、拡張など
  - **物理的視点（運用・オペレーション担当者）**
    - システムの「実物」について見る
    - 必要不可欠なインフラを調達し、デザインし、構築し、統合し、テストし、展開する
    - システムをサポートし、ソフトウェア構造を物理的構造にマッピングする
    - コンピュータ、ネットワーク、ルーター、機械的なシステム、センサー、配線など

ACDM（アーキテクチャ中心設計手法）

アーキテクチャ中心設計手法（ACDM）とはカーネギーメロン大学で提唱している手法であり、様々なソフトウェア開発手法へ適用できます。
ACDM(The Architecture Centric Design Method：アーキテクチャ中心設計手法)は、アーキテクチャ要求を集め、それらを整理、分析し、アーキテクチャをデザインし、評価
し、そして、それらをイテレーティブに洗練していきます。

ACDMの一般的なステージ



これからは、システム開発者（提供するITサービスを具現化する）は動的視点・静的視点・物理的視点の三つの視点をバランスよく考慮して設計作業を行わなければなりません。
これを、実現する手法としてアーキテクチャ中心設計手法（ACDM）があります。

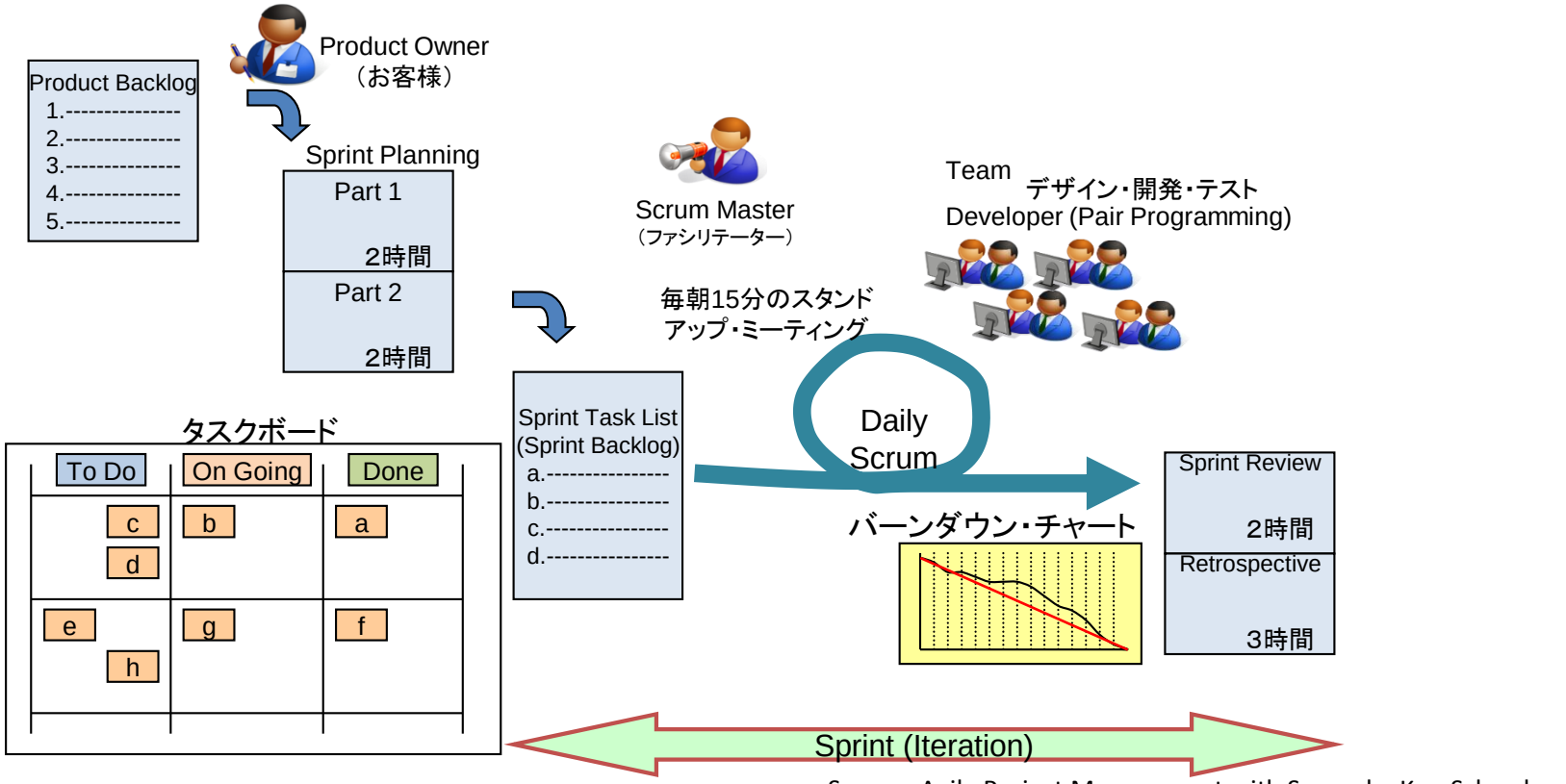
- ACDMはイテレーティブにデザインを開発することに主眼を置きます。
- ACDMの重要な信条は、最初のアーキテクチャデザインにこだわらず、むしろ素早く最初のアーキテクチャを作成し、技術的な課題を発掘するために評価し、アーキテクチャを洗練させていくことです。

システムの開発

アジャイル開発（Scrum）

PANDA（Poor And Non-Disciplined Agile）に陥らない事が大事

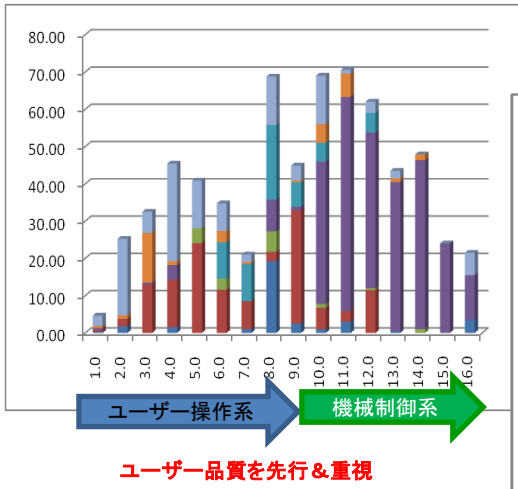
タスク粒度を細かく均一化して、品質の維持と機敏性の確保
Scrumチームのリズムが全体のプロセスのHeart beatになる。（ペースメーカー）



Scrum: Agile Project Management with Scrum by Ken Schwaber

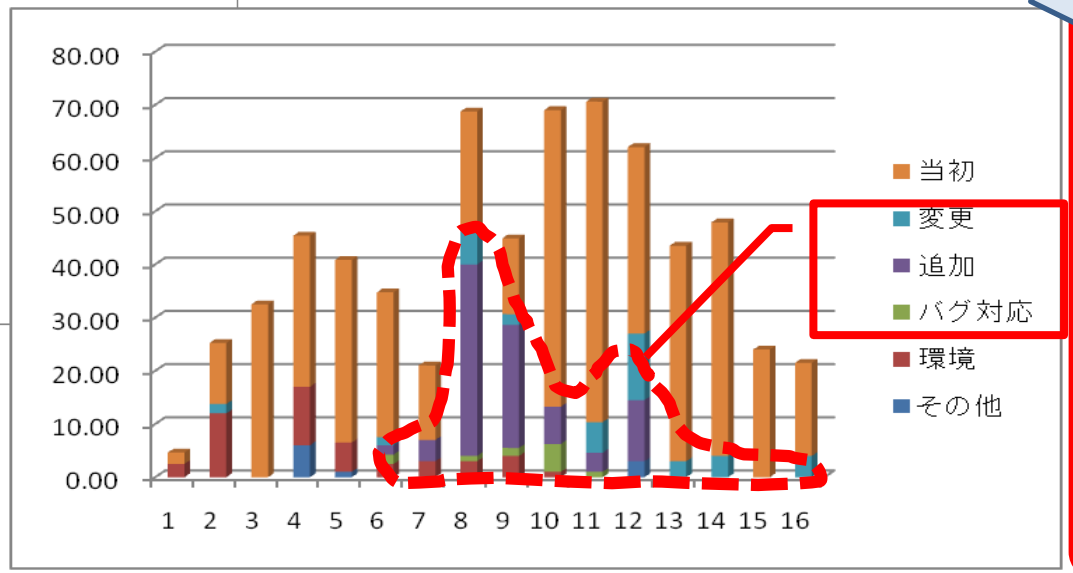
企画 要求 設計 開発 移行 稼働 運用 EOL

毎週の機能別実績時間



ユーザー品質を先行＆重視

毎週の作業要因別実績時間



自工程完結の結果

納品後のバグは二件のみ
全て機器調整に対応する為

タスクの粒度と平準化（実績）

	見積総時間	実績総時間	タスク総数	見積粒度（H）	実績粒度（H）
第1週～第7週　イテレーション　**ユーザー操作系**が主な機能	197.8	211.9	134	1.476	1.581
第8週～第16週　イテレーション　**機械制御系**が主な機能	418.4	452.2	295	1.418	1.533

全く異なった機能実装イテレーションでも同じ粒度で平準化している

システムの移行

このフェーズでは、もちろんこの前のシステム開発において自工程完結で作業ができていれば、プログラムの品質は担保されているはずであるが、ユーザーにソリューションとして提供されるシステム全体としての確認＆テストは必要である。このフェーズでは下記の二点が主要な活動となる。

① ソリューションとしてのシステム全体での運用テスト
② ユーザーに必要なドキュメントの整備

① ソリューションとしてのシステム全体での運用テスト

ユーザーのオペレーション・シナリオに基づきソリューションとして運用可能であるかの確認をユーザーに確認して貰う。またこのテストが最終のユーザー受入テストとなる。
これはユーザーのビジネス・プロセス上、新たに提供したソリューションとして機能できるか？スムースに運用できるか？という点を確認する事である。

② ユーザーに必要なドキュメントの整備

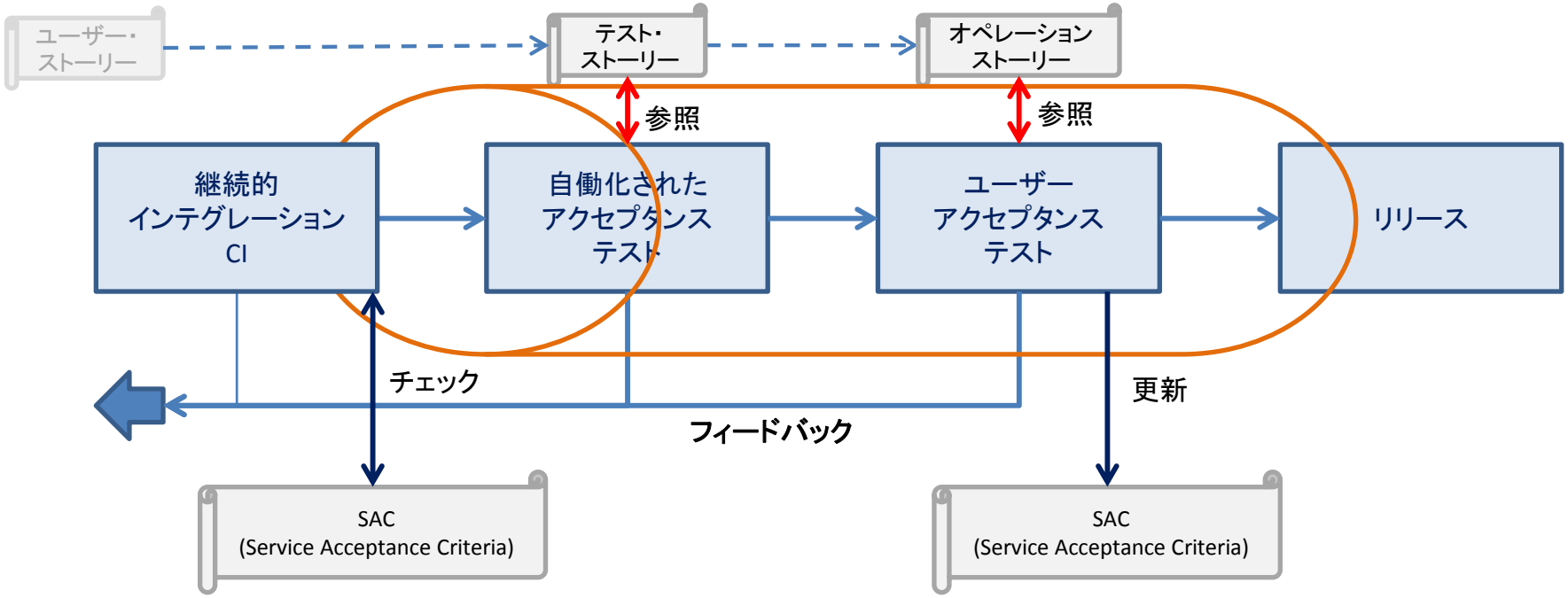
最終的にユーザーへ提供されるドキュメントを整備する。ユーザーに必要と思われるドキュメントは、

- メンテナンス・マニュアル
- SAC（Service Acceptance Criteria）とシステム運用マニュアル（含む障害回復）
- 導入ガイド
- ユーザー・マニュアル （操作マニュアル＆ガイド）

- Build, Test, Deployment**のプロセスの手順化＆自働化**（ITIL®の視点）
  Delivery Ecosystemの検討
- テスト終了後にSAC(Service Acceptance Criteria)の見直し

ディプロイメント・パイプライン

継続的インテグレーション（CI）は、開発チーム側の作業
この移行フェーズから運用側へのバトンタッチの作業を継続的に実施する
（継続的デリバリー）
基本的にDevOpsのプロセスを整流化する為にプル型のプロセスを構築する
必要がある。



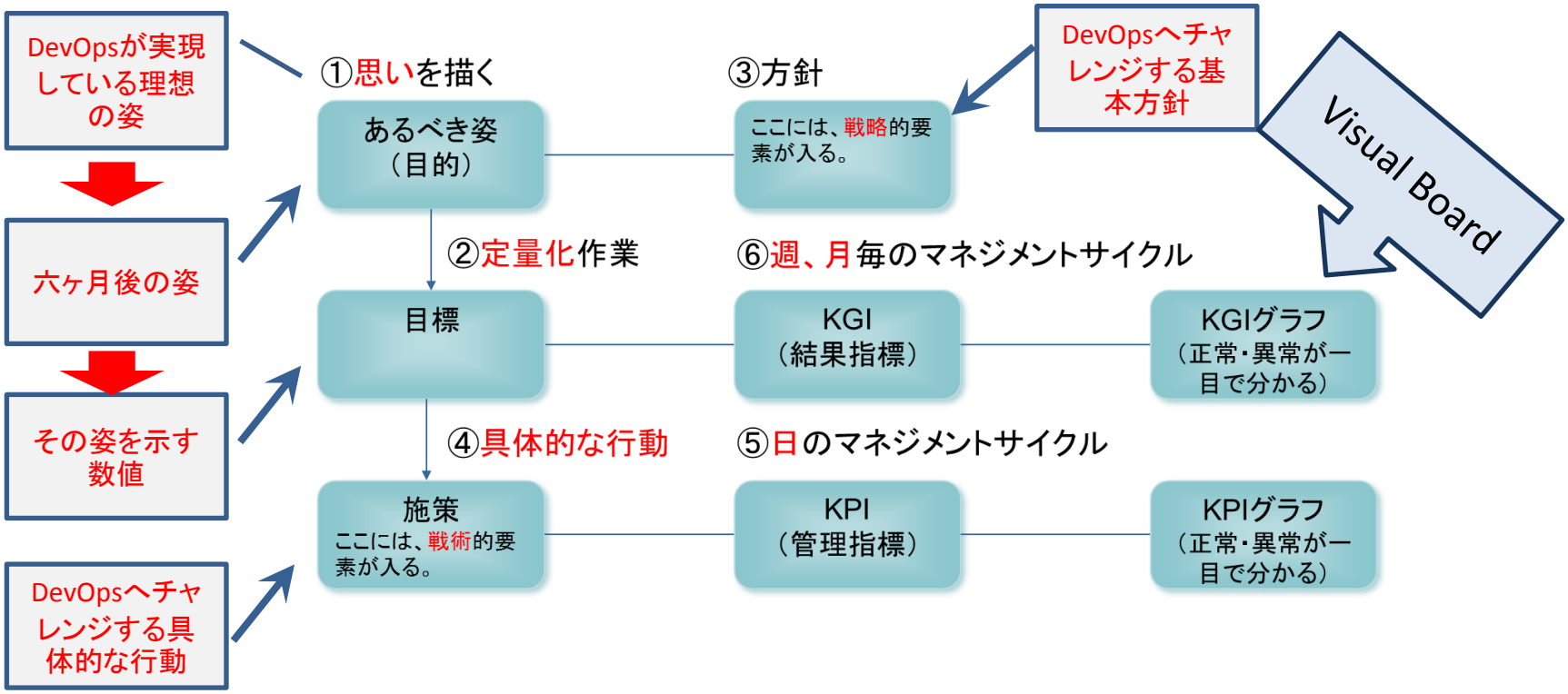
CD: Continuous Delivery by Jez Humble & David Farley

システムの運用

EOLのモニタリング（Project Charterで定義された要件）

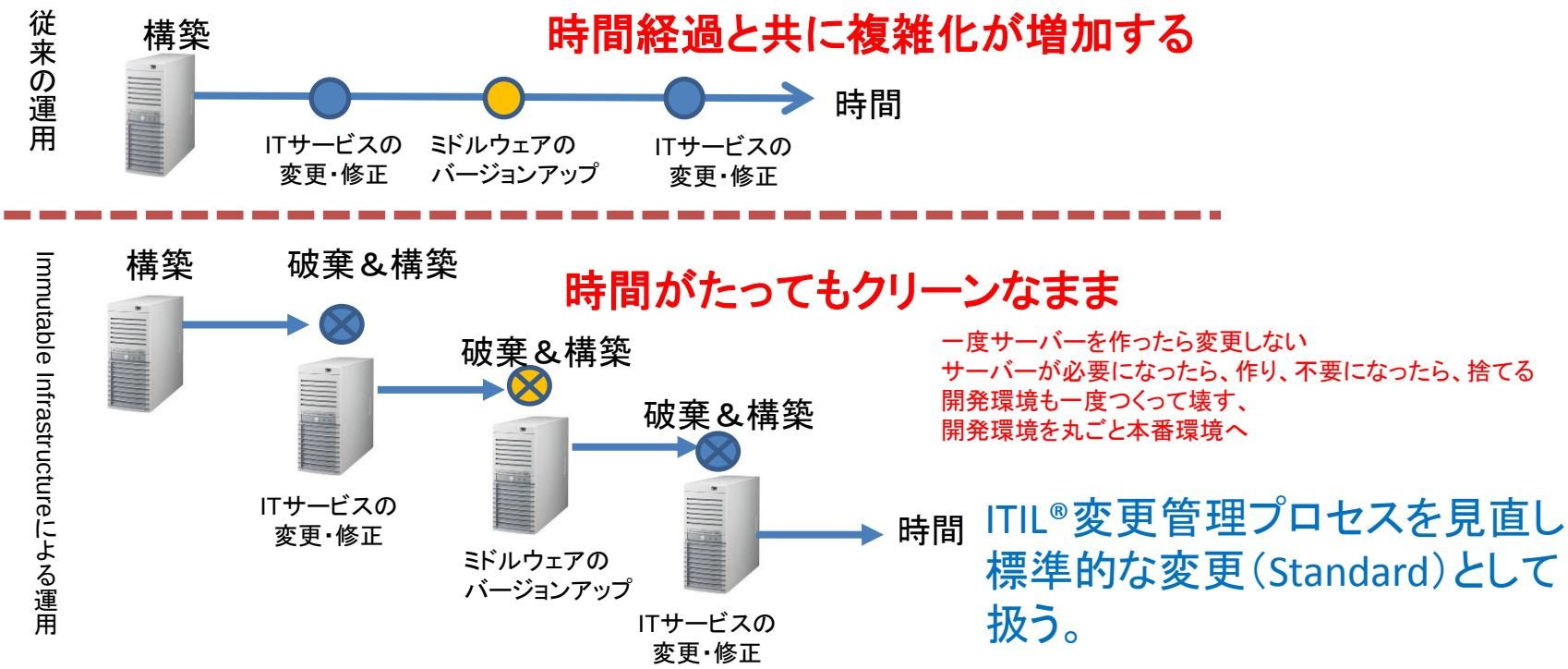
- ◆コスト（見積り運用コスト vs 実績コスト）
- ◆機能（変更要求、バグ、パッチ）
- ◆ビジネス・ニーズの変更（継続性、セキュリティーなど非機能要件）

Base LineからPDCAを回して、パターン化



Base Line（現在の環境設定）から『あるべき姿』の目標とのギャップを把握し、PDCAを回しながら、目標に近づけて行く。
最終的に環境のパターンを作成する（インフラストラクチャのパターン化）

Immutable Infrastructureによる運用



運用側では、従来の仮想化マシンによる仮想化されたITサービスを運用するとなると、その仮想化マシン各々に資源を割り当てるという方法で、物理的なハードウェアを全体を鳥瞰的にタイムリーに管理するといったことが複雑になってしまいます。この両者の観点を上手く折り合いをつけられる技術としてコンテナ技術による仮想化が登場してきています。この技術は簡単に言うとアプリケーションとミドルウェアだけで塊をつくりDockerのコンテナ基盤に載せると、コンテナ基盤をサポートしている様々な環境で運用/オペレーションできることが可能となります。

更に、これまでの一度出来上がった全ての環境を保守・維持するという時間経過と共に複雑化していく運用環境を捨てて、Immutable Infrastructureという破棄＆構築を繰り返す時間がたってもクリーンな運用環境を実現することも出来るようになってきます。

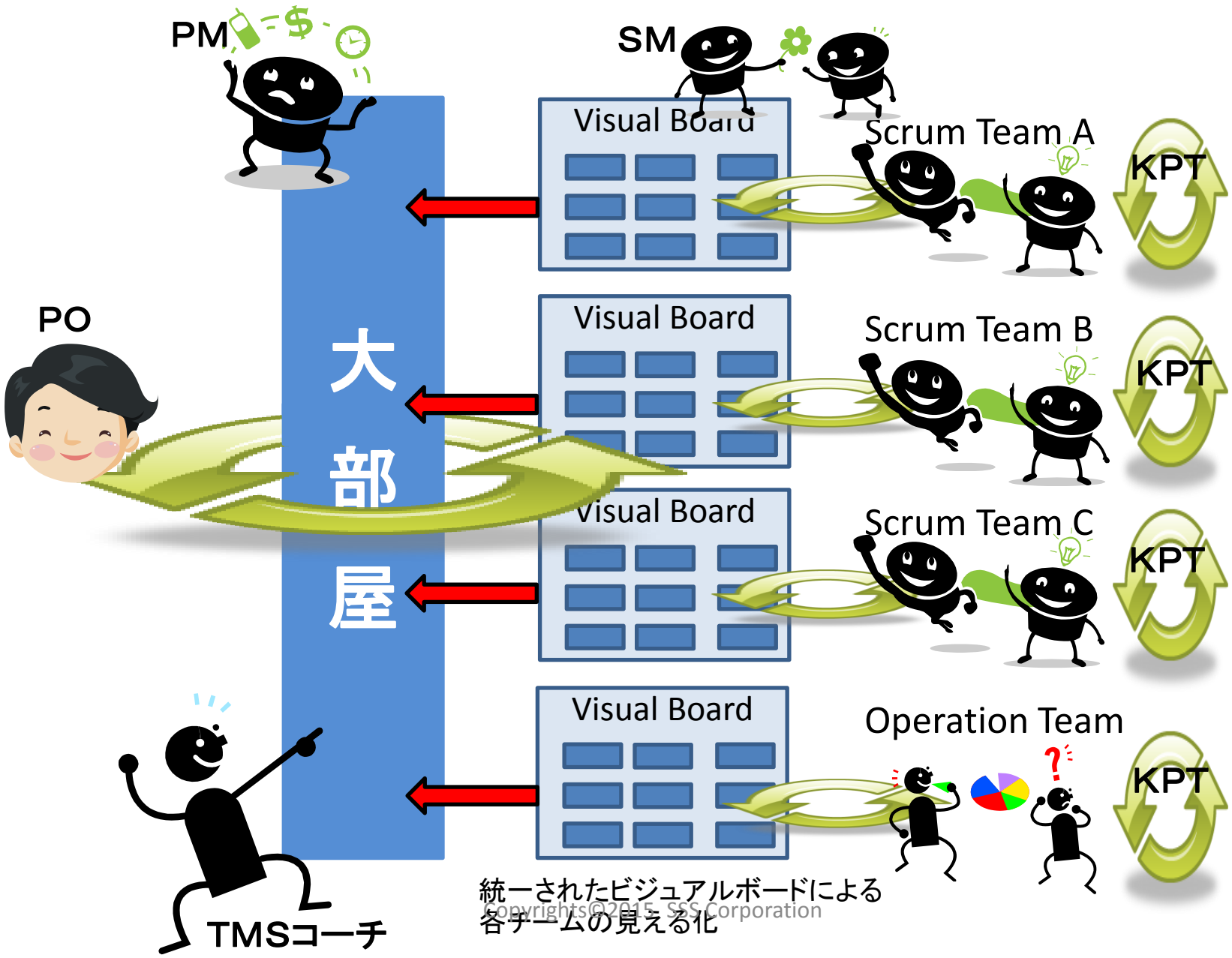
開発から運用までの全プロセスの全体最適

管理体系が異なる性質の仕事で構成されるプロセスの全体最適を狙った一元化をどう実現するのか？

トータルTPS/TMSでの大部屋方式の導入
◆全プロセスでの見える化
◆『一個流し』でのプロセスの整流化

タスクの『粒度（時間単位）』と『均一化』

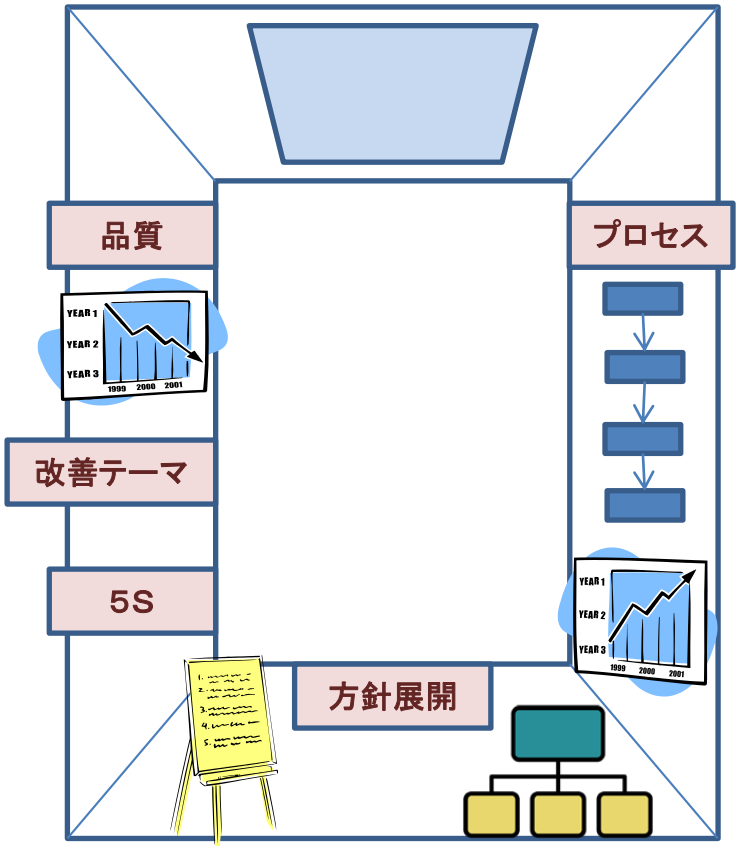
全プロセスでの整流化（澱みなく仕事が流れる）の仕組みの構築とその洗練化の為に常にカイゼン活動が働かなければならない。



統一されたビジュアルボードによる
各チームの見える化

Quickening Visualization System（大部屋方式）

大部屋の模式図



狙い：

- 人と職場（チーム）の活性化を促進する効果が大きい
- プロジェクト全体が俯瞰（見える化）できる
  「見える化」とは、異常が瞬時に解る、気づく事
- Scrum of Scrumの実施会場（場所）

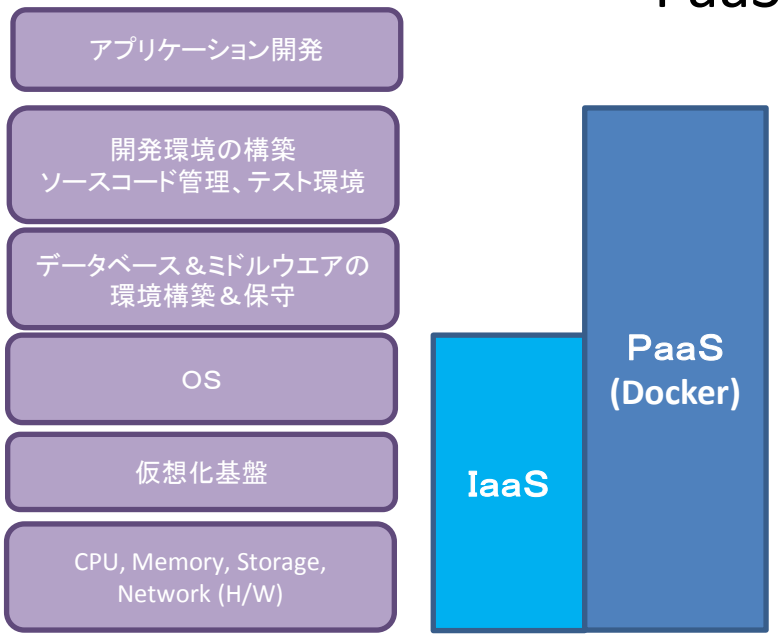
特長：

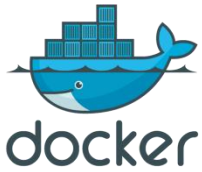
大きなプロジェクトや他部署との連携したチームなど組織横断的な事業に取り組む場合には、『目で見る管理』を大部屋化して行う

大部屋とは、関係者が一つの部屋に集まって課題を見える化し、問題解決を進めていくやり方

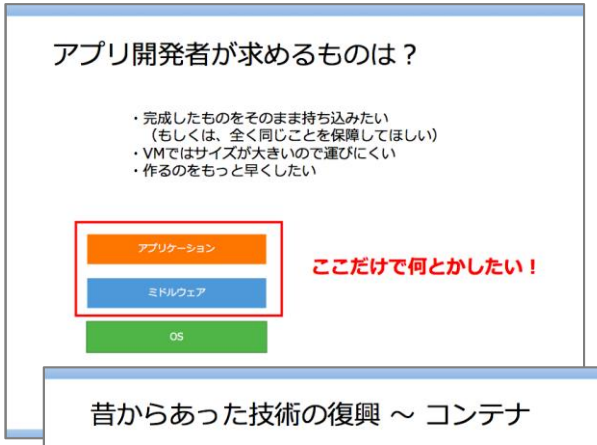
同じ部屋で仕事を行う事でコミュニケーションが活発化しアイデアが沢山でる様になる

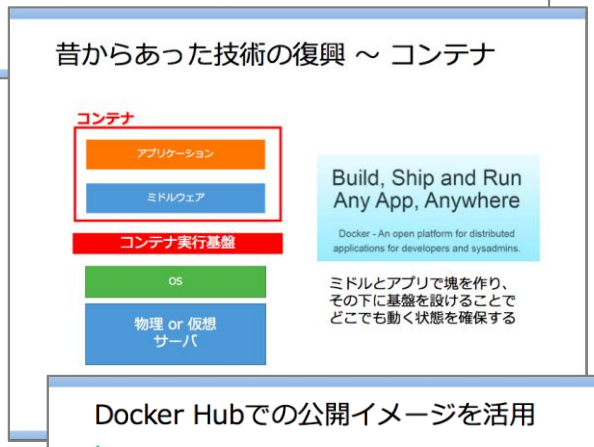
DevOpsのスピードと柔軟性を高めるインフラストラクチャー
PaaS ＋Docker

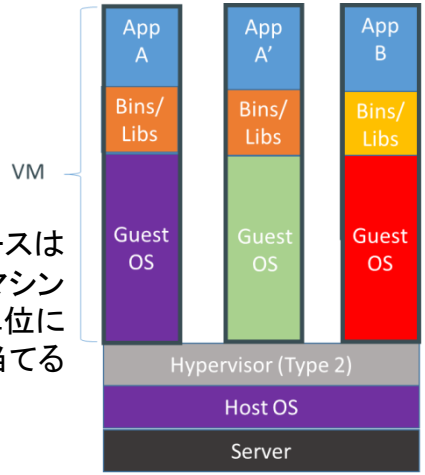




吉田パクえさん資料「捕鯨！詳解docker」より抜粋
http://www.slideshare.net/yuya_lush/docker-42739730

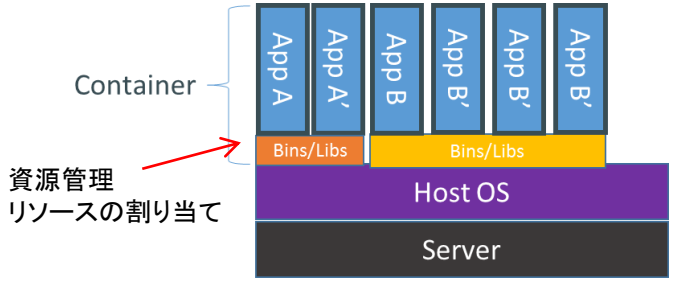






リソースは
仮想マシン
単位に
割り当てる

仮想マシンによる仮想化

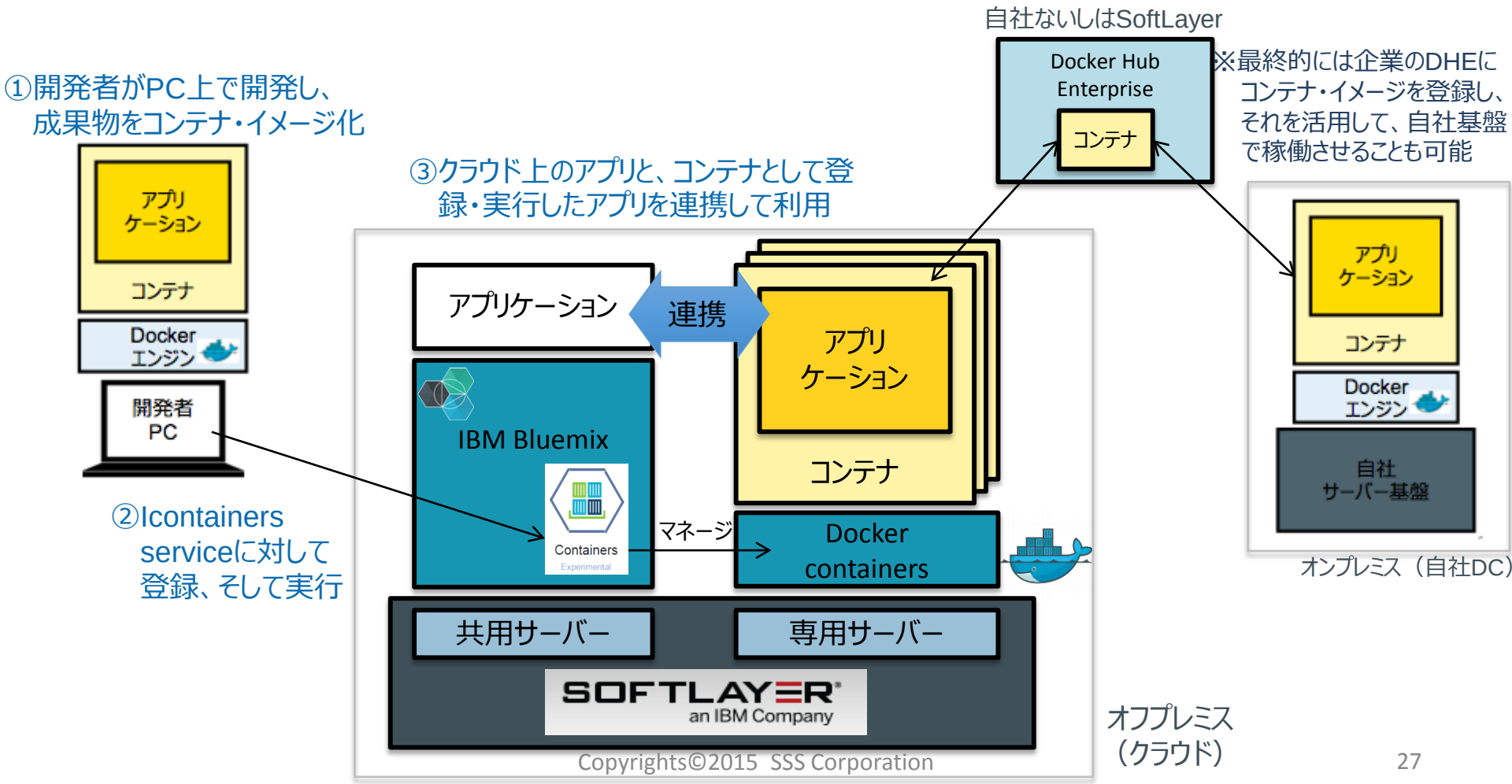


資源管理
リソースの割り当て

コンテナによる仮想化

BluemixでのDockerコンテナー・サービス

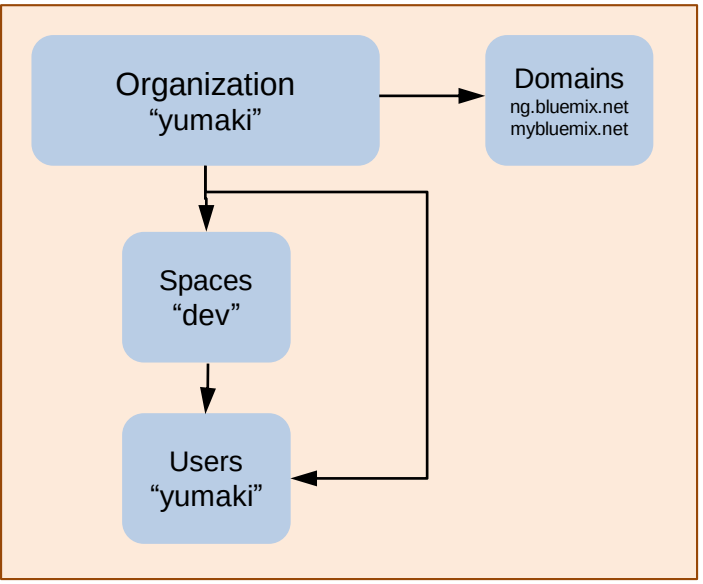
Docker実行環境のセットアップ・運用保守の労力無く、かつクラウドで提供されるサービスに限定されずに、エンタープライズ独自の開発フレームワークや固有の環境に依存したアプリ開発をDockerを活用して行う事が可能。



Bluemixコンテナーでのアカウント管理機能設

Docker Container Image Registry

＜デフォルト＞

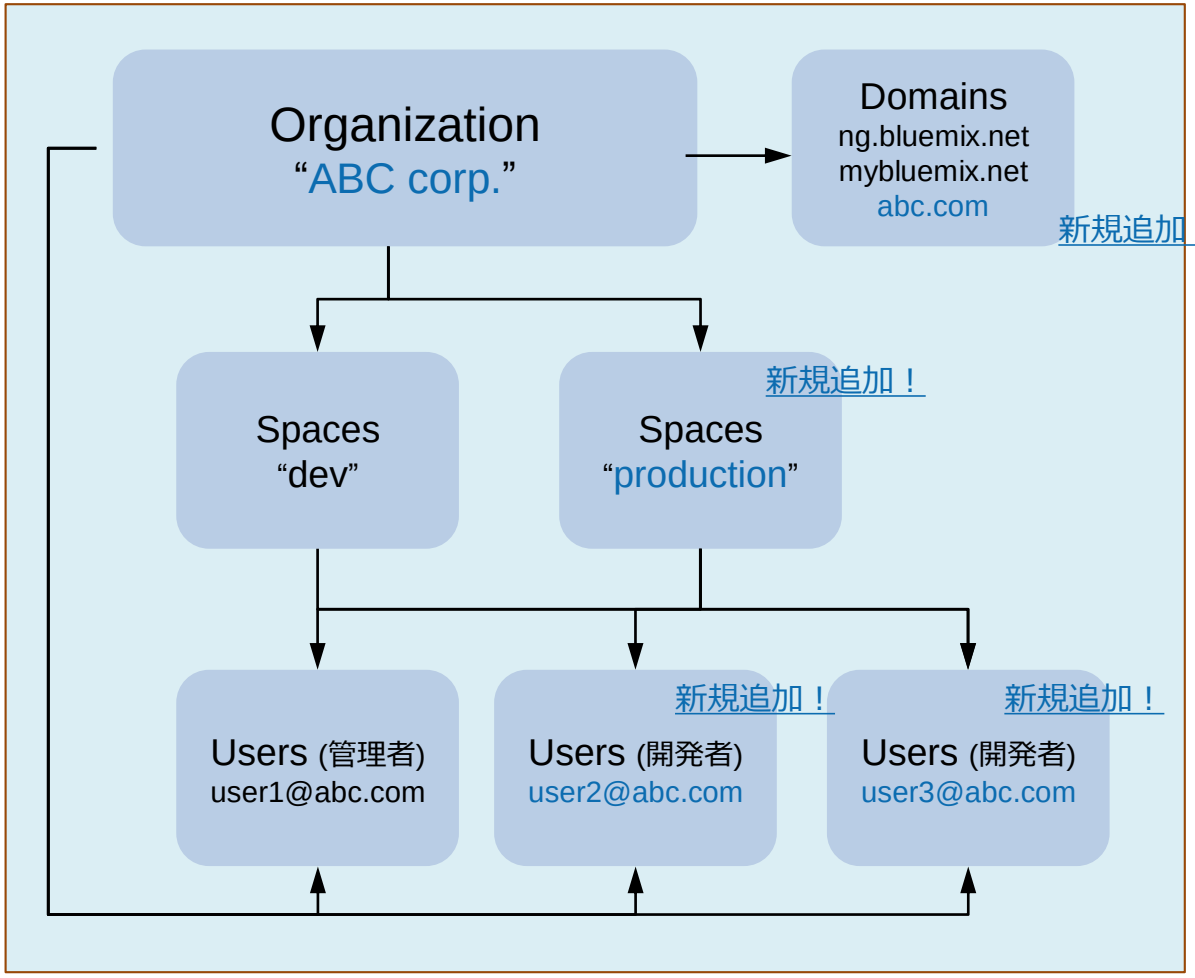


組織的に利用するためには以下の“概念”を提供します。

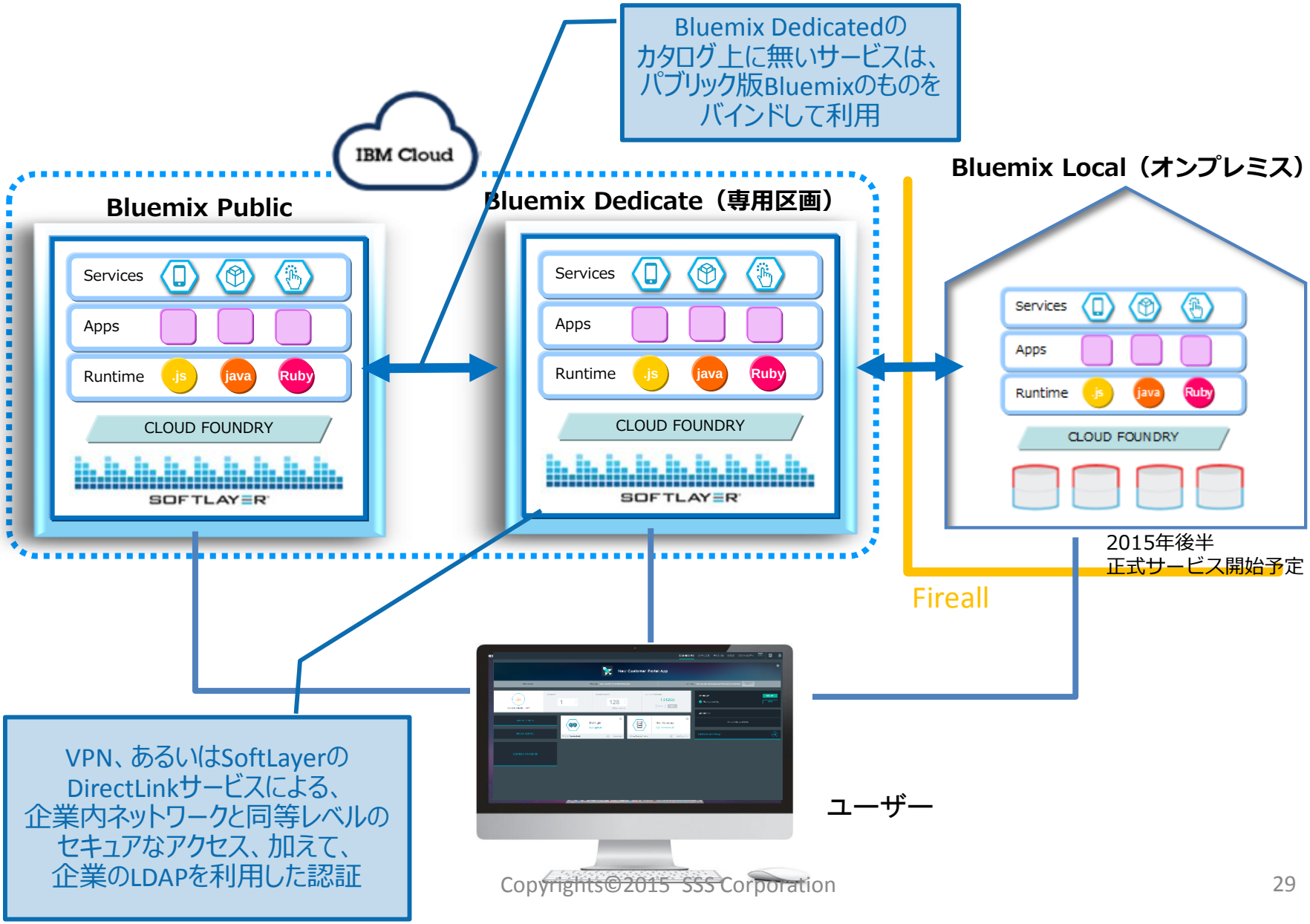
①**Organization**（組織）
そして、Organizationは以下の4つで構成されます
②**Spaces**（作業スペース）
③**Users**（ユーザー）
④**Domains**（インターネット・ドメイン）
⑤**Quota**（リソース容量

※他にも、“Region”（データセンターの選択）がありますが、Organizationを含むこれら構成の、更に上位に位置づけられる概念となっています

＜企業での構成例(Bluemix)＞



Bluemixでのインフラストラクチャーの柔軟性

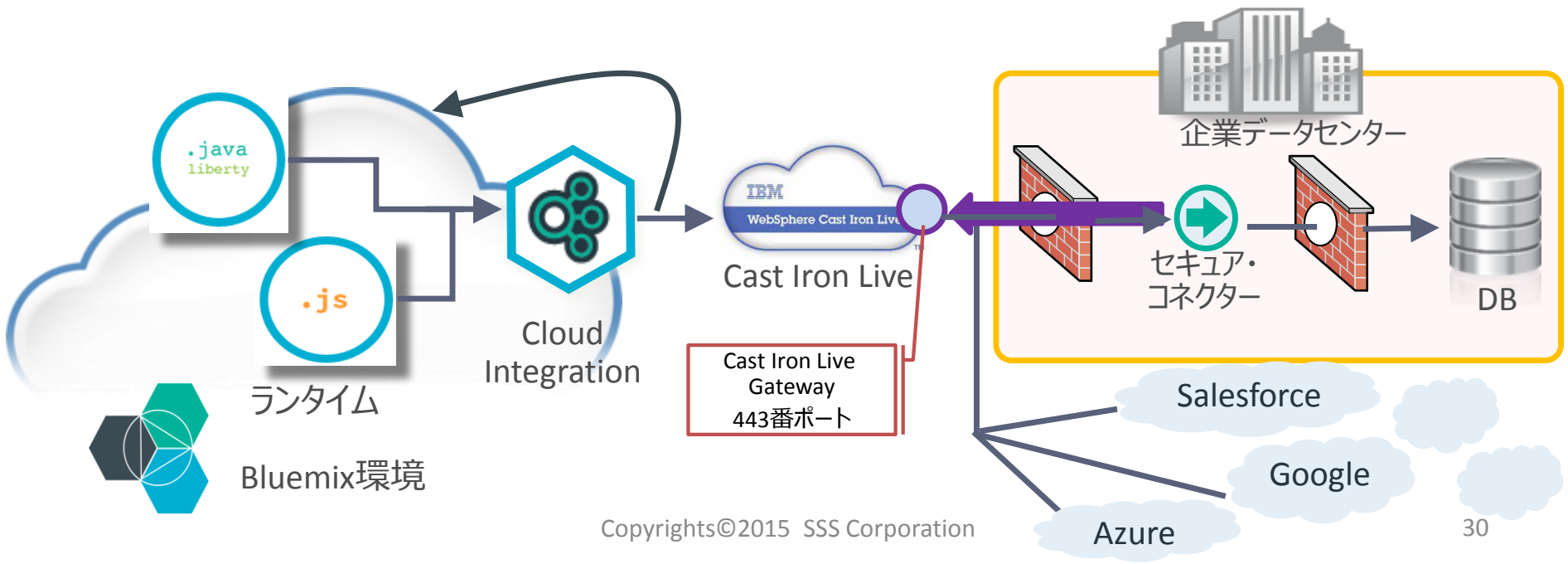


既存システムとの連携 “Cloud Integration”

- Bluemix上で稼動するアプリケーションから、既存システムへアクセスする手段を提供
  - 既存システムに対してHTTPベースのRESTfulなアクセスを実現

※セキュア・コネクター

- Proxyとして動作し、この接続を利用してCloud Integrationからの要求をエンドポイントに送信
- セキュア・コネクターからCast Iron Liveに対するアウトバウンド通信のみ、ファイアウォールで許可すれば可



まとめ

ビジネスのスピードを牽引するジャストインタイムでのITサービスを提供する為には、、、

- ◆ 変化に対応する機敏なアジャイル開発（タスクの粒度）
- ◆ 安定したITサービスの運用を担保するITIL® V3（環境のパターン化）
- ◆ 柔軟かつ機敏な稼働環境を作るPaaS + Dockerのクラウド環境

以上の三点が必須の要件です。

あとは、
開発から運用までの全てのプロセスの整流化と全体最適を追求するカイゼン活動が必要です。

DevOpsプロセス成熟度モデル

DevOpsは、ソフトウエアの開発～運用現場でのプロセス改善、品質改善活動です。

- **S-1(ステージ1) 自己管理　Team Building**
  - DevOpsの知識・技能を習得し、自己管理の為の分析力が働くチーム
    開発と運用の一体化（目的の共有）
- **S-2 (ステージ2) プロセスの見える化　End to End Traceability**
  - 全プロセスに渡るチームの活動の『見える化』、活動のログを管理し改善に繋がる
- **S-3 (ステージ3) プロセスの安定化　Stabilize Service Providing Process**
  - 顧客、システムのサービス品質を理解し、その品質を満たす。
    （顧客第一での品質への拘りがチームに出る）
- **S-4 (ステージ4) 学習する組織　Learning Organization**
  - 品質、生産性、納期（リードタイム)を意識し改善が進む学習する組織（チーム）
- **S-5 (ステージ5) 自律した組織　Anticipated ROI & Failure Tolerant Organization**
  - 完全に自律した問題解決適応力のある組織（チーム）

DO001：

失敗しないDevOps（管理職編）
－DevOps概説からパラダイムシフトまで

株式会社 アイ・ラーニング
i-Learning

・開発と運用チームとで食い違いが起きていませんか？
・開発と運用のスピード感は合っていますか？
・クラウド時代に対応した開発や運用になっていますか？
・現在の仕事の進め方、やり方に疑問を感じませんか？

DevOpsにモヤモヤしている方、DevOpsを本気で実践したい方へ！

この解決策を自ら考えられるようになります！

この研修は、今までの考え方からパラダイム・シフトを起こさせます！

ワールドカフェ・スタイルでのチーム・ディスカッションを実施し、相互の見識を広げるとともに、これまでの概念を超えた議論を進めます。

自ら考え、他人ごとから、自分事へと考えを変えていくコースです。

コース概要

・対象者：開発系および運用系の管理職および管理職相当の方
・料金：35万円(税込み37.8万円)/人
・定員：最大15名/クラス
・期間：半日(土曜日) x 全6回 (2週間毎に開催。計3ヶ月)
2015年 第1回目：5/16、5/30、6/13、6/27、7/4、7/18

同じ会社から開発系・運用系それぞれ参加されることをお薦めします。

2名ペアで参加される場合は、各5万円引きの1名30万円となります。
複数名以上でお申込みの際は、事前にお問い合わせください。

コース概要

1. DevOps概要
   - 開発～ディプロイ～運用のパイプライン
   - プル型のシステム提供
   - 開発～運用環境としてのクラウド（PaaS+Docker+Kubernetes）
2. 要求の確定
   - ユーザー・ストーリー
   - テスト・ストーリー
3. システムの設計
   - アーキテクチャー中心設計（ACDM）
   - 工程設計
4. プロジェクトの見える化/チームビルディング
   - TMS概説
   - タスクボード、KANBAN
   - ビジュアルボード
   - 振り返り（KPT）
5. 品質の管理
   - フィードバック・ループの検討
   - 自工程完結のオペレーション
6. 運用の管理（ソフトウェアのフィールド保障）

毎回、次回のテーマに関する宿題を提示します。次回は、その内容を基にワールドカフェ・スタイルでのチーム・ディスカッションや発表を実施し、各テーマの深堀りをします。
単なる座学ではなく、本気でDevOpsを考える人のための研修です。

お申込みはWebから

https://www.i-learning.jp/products/detail.php?course_code=DO001

ご清聴ありがとうございました。