

# ITモダナイゼーションソリューション

XXXX

## 資料作成の考え方

業務イノベーション & ビジネス即応

5年、10年以後のビジネスを目指す

プロジェクトマネジメント

統括管理

システム設計

エンドユーザー業務イノベーションをスムーズに行う

システム開発のプロセスとリスク管理

スクラムで段階的に機能を追加できること

リスクを軽減し、価値を積み上げていくこと

コストコントロール

作業標準化になって 自動作業ツールを活用し、モノリシックよりコストを6割以上減少すること

## 目次

ITモダナイゼーションの基本  
現行システムの調査  
再構築  
開発プロセスとマネジメント  
リスク対策  
作業自動化ツール  
ケーススタディ

# ITモダナイゼーションの基本

## ITモダナイゼーション

ITモダナイゼーションとは、古くなったIT資産（レガシーシステム）を最新技術に対応させ、近代化を図ることを指します。

ITモダナイゼーションソリューションは お客さまのビジネス、現行システム資産の特性、課題、要求仕様を整理し、最適なITモダナイゼーション手法の検討から、次世代のビジネス即応型システムの構築・保守までを支援します。

## 次世代のビジネス即応型システム

グローバル化するビジネスへの対応


企業内、企業間、企業と消費者間のネットワークで展開されるビジネスを支援

業務プロセスの変革に着目して業務間連携と企業間連携の容易な実現を支援

お客様のシステム構築に携ってきた経験と実績を基に、最適なサービスや製品を組み合わせ実現

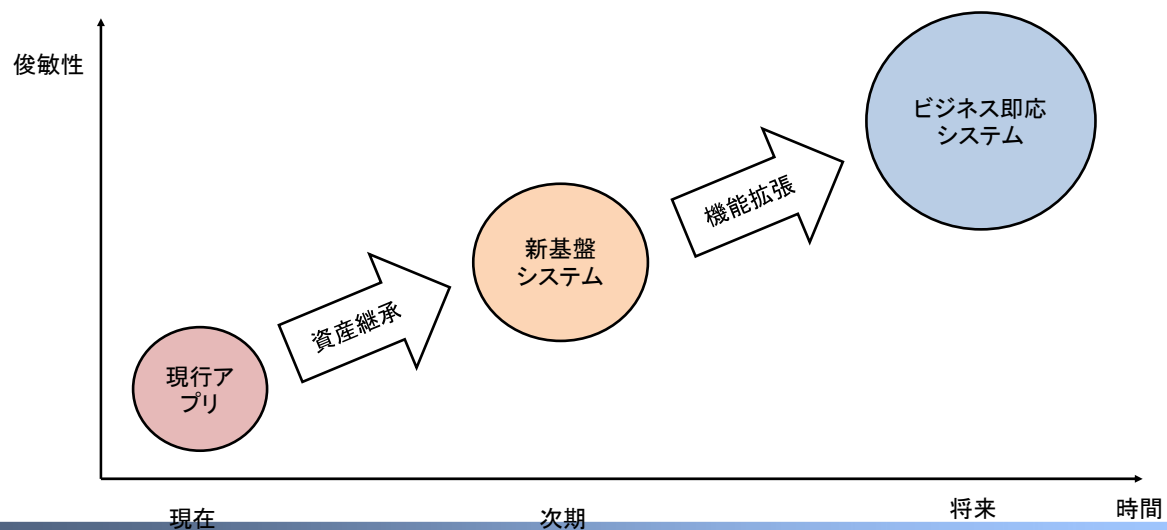
## ビジネス即応型システムのメリット

急激に変化するマーケティングへの素早い対応  
経営と業務プロセスの視点から体系化したビジネス・ソリューションの利用  
アプリケーション間連携の容易な構築  
最適でトータルなビジネス・ソリューションを短期間で構築  
オープン環境で信頼性の高い安定したビジネス・ソリューションの利用  
異機種環境での情報システム間の相互運用  
最適なプラットフォームの選択  
統合的なセキュリティ環境の構築



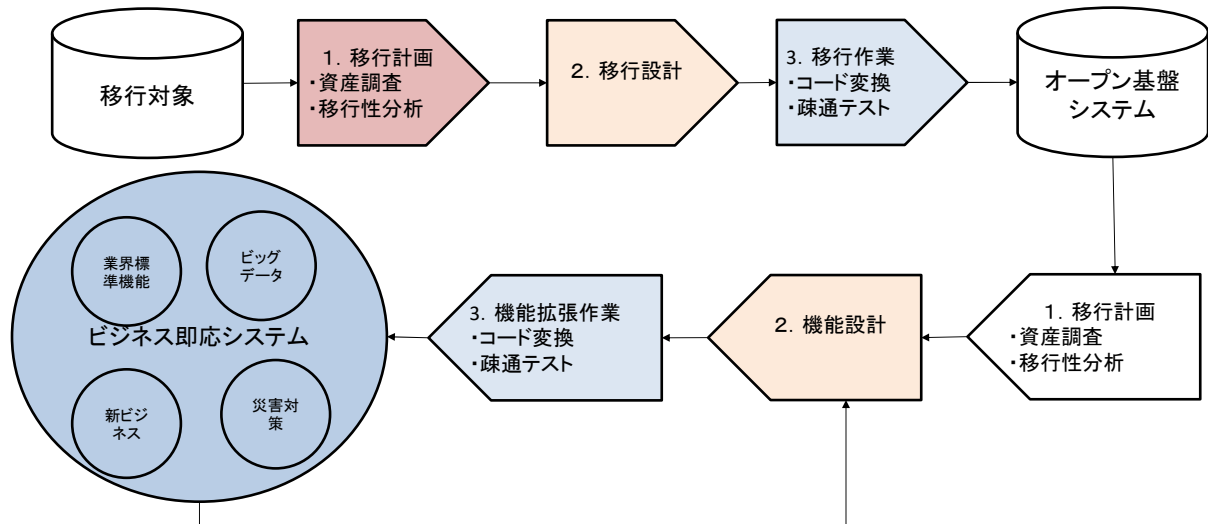
マイクロサービスによっ  
て修正

## 戦略的な機能を段階的に拡張



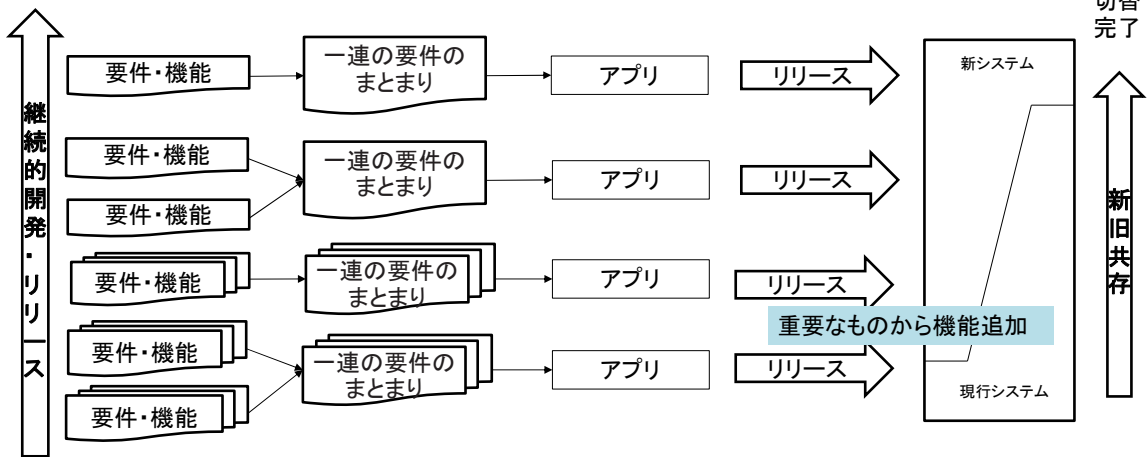


## ITモダナイゼーションのプロセス



## ITモダナイゼーションのプロセス

時間軸



## 現行システム調査

### リファクター

機能をそのままに現行のコードを整理する

性能や保守性を向上させる為に、現行アプリケーションの設計や実装を見直すことです。マイグレーションの文脈においては、必要な資産の範囲を見極める「棚卸」が重要です。

### リドキュメント

現行システムを可視化、ドキュメントを再整備

現行システムの資産分析やヒヤリングを通じて、現行システムの仕様、設計情報を可視化し、ドキュメントを整備する技術です。

## 再構築

### リホスト

既存システム資産をそのまま利用する方式

互換性のあるミドルウェアやエミュレーションを利用して既存アプリケーションをそのまま利用する方式です。移行費用を押さえつつ短期にインフラを刷新したい要望に応えることができます。

### リライト

変換ツールでコードを他言語に書き換える方式

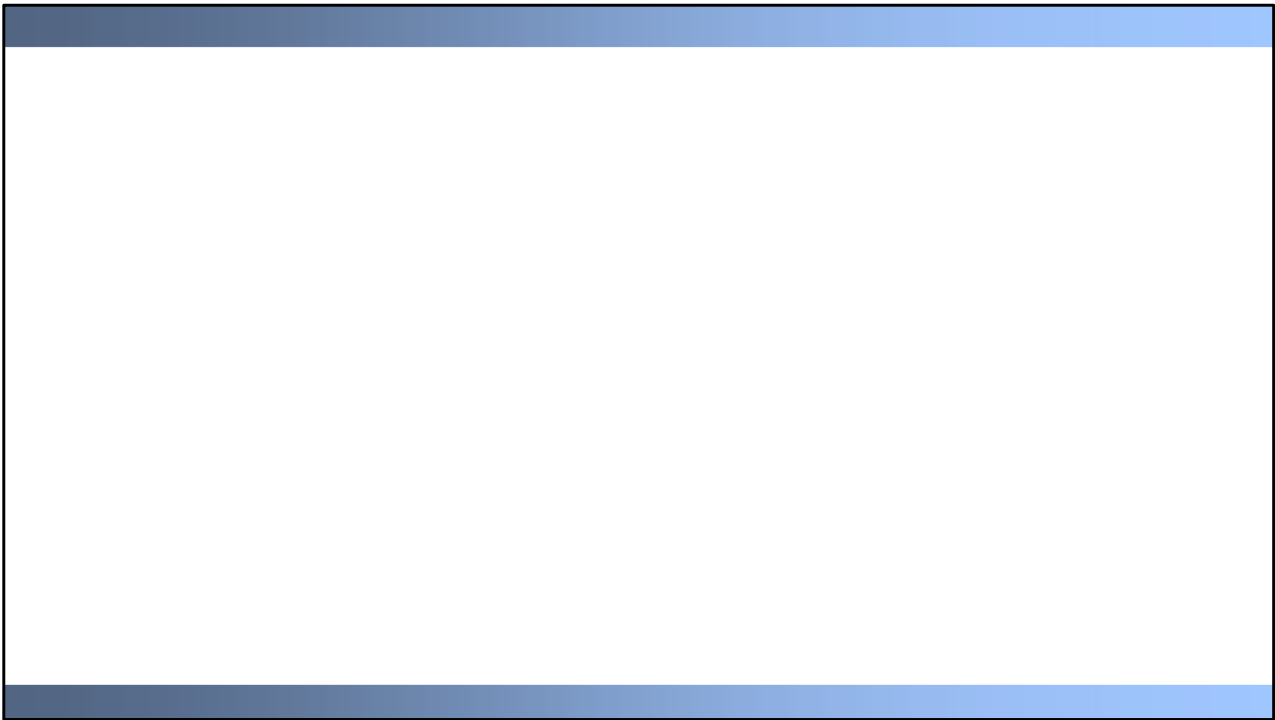
既存のプログラムを変換ツールで他言語に書き換えて、移行を実施する方式です。開発・保守の環境含めを新しい言語に統一したい要望に応えることができます。

### リビルド

既存システムの業務仕様を再実装する方式

既存システムの仕様を生かし、再実装する方式です。移行方式としては最も自由度が高く、非機能要件に対し、最適な設計／実装の選択が可能ですが、最もコストがかかります。

**全体工程計画 参照P143**



# 現行システム調査

## 現行システム調査の目的

再構築において拠り所となる現行システムの状態を正確に把握するために、プログラム仕様の可視化と事実ベースの課題抽出、現行システムの調査・分析を行う。

参照

[「システム再構築を成功に導くユーザガイド～ユーザとベンダで共有する再構築のリスクと対策～」](#)（独立行政法人 情報処理推進機構(IPA)2017年発行）



## リファクター

### 目的

性能や保守性を向上させる為に、現行アプリケーションの設計や実装を見直すことです。マイグレーションの文脈においては、必要な資産の範囲を見極める「棚卸」が重要です。

### 方法

#### 棚卸

システム再構築では資産を整理する機会なので棚卸は重要である。

後続工程(設計～テスト、保守)も含めたプロジェクト全体のコストを削減できる。

## リファクターの技術:スリム化

### 目的

不要なプログラムや類似性の高いプログラム(コードクローン)をツールの活用によって洗い出し、削除・整理します。

## リファクターの技術:スリム化

スリム化

## リドキュメント

### 目的

現行システムの資産分析やヒヤリングを通じて、現行システムの仕様、設計情報を可視化し、ドキュメントを整備してシステムのホワイトボックス化を図ります。

### 計画の検討材料

業務フローやジョブフロー、CRUD図\*等、可視化で作成したドキュメントは、移行計画やテスト計画の検討材料になります。

### 問題分析や影響分析の支援策

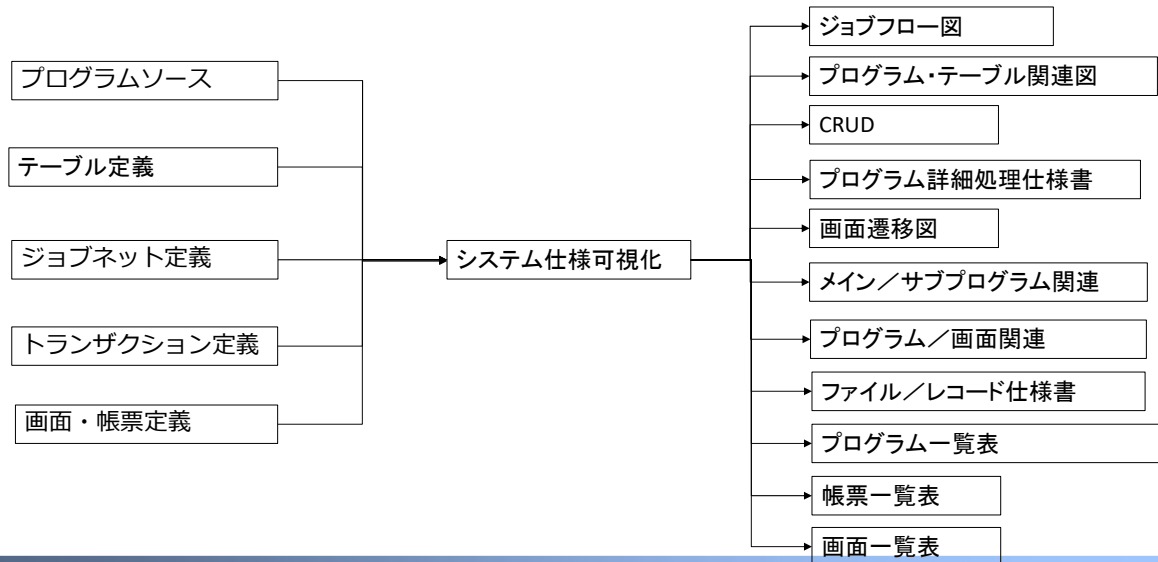
テストでの問題発生時に既存の実装を確認する為に、既存のプログラムの可視化結果を利用することができます。

問題への対応方式を検討するにあたって影響分析に用いることができます。

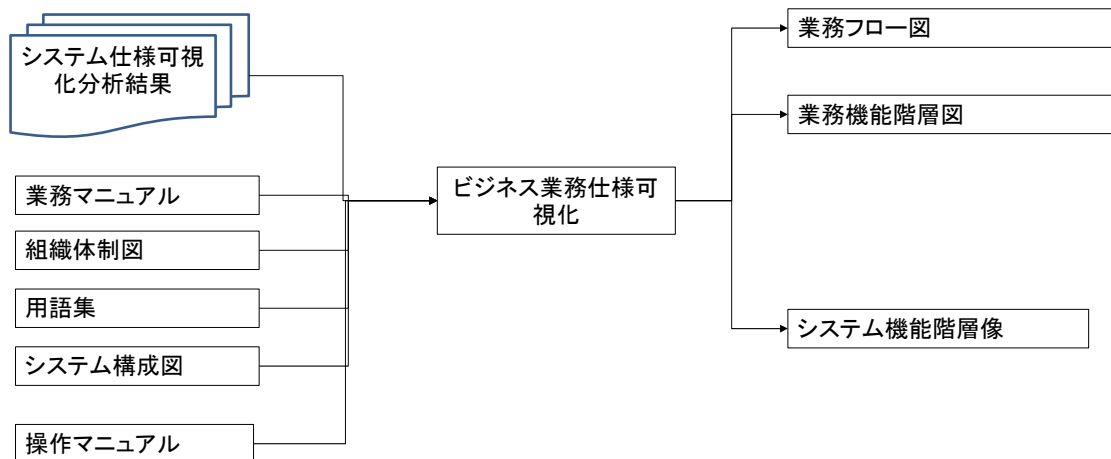
### 再構築の前提としてのリドキュメント

特にリビルド型のマイグレーションでは、ドキュメントが保守されてない場合、必須のタスクとなります。その他の場合でも、ツールで作成した仕様情報を用いることにより作業の効率を上げることができます。

## リドキュメントの技術



## リドキュメントの技術



# システム再構築

## マイクロサービスの基本方針

小規模なサービスを組み合わせてアプリケーションを開発すること

各サービスは、独立したプロセス上で稼働すること

各サービスは、RESTなどの軽量プロトコルで通信すること

各サービスは、自動的に、それぞれ個別にデプロイできること

各サービスは、それぞれ異なるプログラミング言語で実装可能であり、かつ、それぞれ異なる永続データストアを利用可能であること



## マイクロサービスの9つの特徴

サービスのコンポーネント化-コンポーネントは独立して交換・更新可能なソフトウェアの単位である。  
ビジネス中心組織-開発運用チームは技術や開発工程によるチームではなく、ビジネスを中心に機能横断型のチームが編成されている。  
プロジェクトではなくプロダクト-チームは開発完了とともに解散するプロジェクトモデルではなく、製品のライフサイクル全体に責任を持つ。  
スマートエンドポイントとダムパイプ-メッセージ通信は軽量かつシンプルであること。エンドポイントに高い機能を持たせ、通信はダムパイプ(メッセージのルータとしてのみ動作する単純な機構)であること。

## マイクロサービスの9つの特徴

分散統治 - 各サービスは独立したチーム、開発言語、ツールでアプローチする。

分散データ管理 - 概念モデルに関する分散だけでなく、データストレージも分散(サービス間で独立)する。

インフラストラクチャの自動化 - テスト自動化/継続的デリバリー/継続的インテグレーションの導入

障害の設計 - 個々のサービスはいつでも失敗する可能性があるため、互いに障害を迅速に検出し、可能であれば自動的にサービスを復元できることが重要。

進化的な設計 - 頻繁に、迅速に、適切に制御されたソフトウェアの変更・廃止・構築を行うことができること。

## マイクロサービスアーキテクチャのメリット

俊敏性

イノベーション

品質

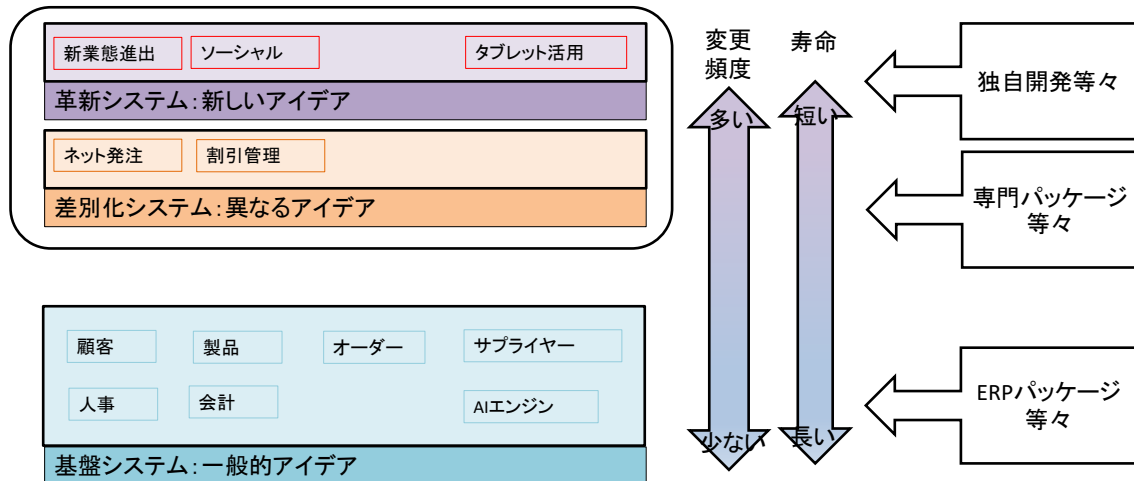
拡張性

適切に非干渉化されたサービスは、水平方向に、それぞれ単独にスケールすることが可能です。

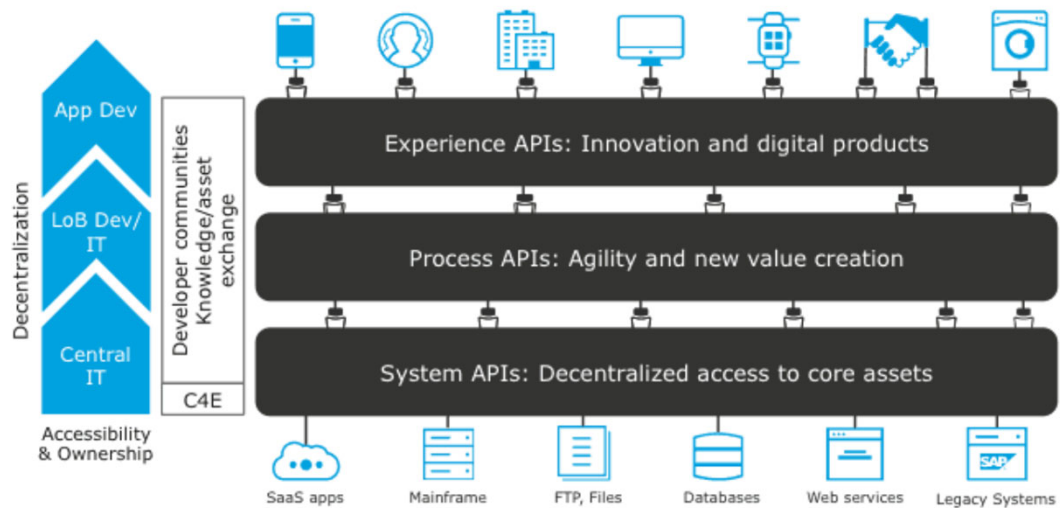
可用性

障害分離の実装を容易にします。

## マイクロサービスのペースレイヤー



## DX対応のアーキテクチャ



## SOA 設計の 10 の原則

相互運用性 - サービスは、加入者がサービスを使用できるようにする標準を使用する必要があります。これにより統合が容易になります。

疎結合 - サービスは相互の依存関係を最小限に抑えます。

ナレッジカーテン/サービスの抽象化 - サービスは、ロジックをカプセル化し外部から隠します。

リソース管理/サービスの再利用性 - ロジックは、再利用を最大化するようにいくつかのサービスに分割されます。

サービスディスカバリ - サービスは見つけられるべきであり、見つけることができます(通常はサービスレジストリにおいて)。

構造的独立性/サービスの自律性 - サービスは、サービス自身が消費するあるいは依存するリソースを制御する必要があります。

サービス構成/構成可能性 - サービスは、大きなタスクを小さなタスクに「分割」します。

粒度/サービスステートレス性 - 理想的には、サービスはステートレスでなければなりません。

サービス品質 - サービスは、サービスプロバイダとクライアント間のSLAを順守します。

高い凝集性 - サービスは、理想的には単一のタスクに対応するか、同じモジュールの一部として類似のタスクをグループ化する必要があります。

## Docker、Kubernetesを取り巻く環境の変化

2013年

Docker社Dockerリリース

2014年

Google社発表全てのソフトウェアがコンテナ上で稼働（毎週20億個起動）⇒ Google社では当たり前の技術

Google社Kubernetesオープンソースとして公開

Docker社Docker Swarm発表（Kubernetesの対抗馬）

2015年

コンテナ標準団体「Open Container Initiative(OCI)」発足

Cloud Native Computing Foundation(CNCF) 設立

⇒Kubernetesの開発主体となる（オープンテクノロジー、ベンダーロックイン無し）

2017年

OCI コンテナランタイム、コンテナイメージ標準化

Docker社Kubernetes統合発表⇒ Kubernetes 事実上標準

Microsoft、Google、AmazonがKubernetesのマネージドサービス開始

コンテナランタイム、Kubernetes間通信標準化

## ビッグデータサービス



## AIエンジン

機械学習(マシンラーニング)は「膨大なデータを基に分析をおこない、精度の高い予測・判断を実現する技術・研究分野」のことです。

教師あり学習 入力値と正解がセットになった学習データをコンピューターに与えて、学習させる方法です。

教師なし学習 正解データが与えられない学習方法で、データの傾向を分析することが目的となります。

強化学習 試行錯誤を通じて、報酬(評価)が得られる行動や選択を学習します。

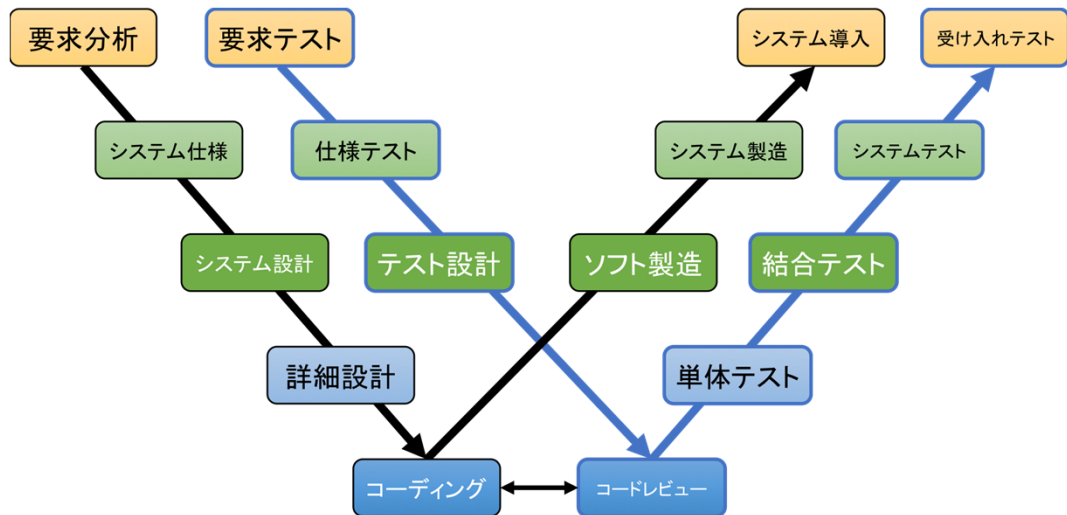
ディープラーニングとは「深層学習」とも呼ばれており、人間が無意識のうちにおこなっている行動をコンピューターに学習させる技術のことです。機械学習の中の「教師あり学習」の一部です。基本的にニューラルネットワーク(人の神経細胞=ニューロンの仕組みを真似たシステム)が基盤になっています。

## 業務自動化

ノーコード・ローコード開発について

# 開発プロセスとマネジメント

## Wモデル



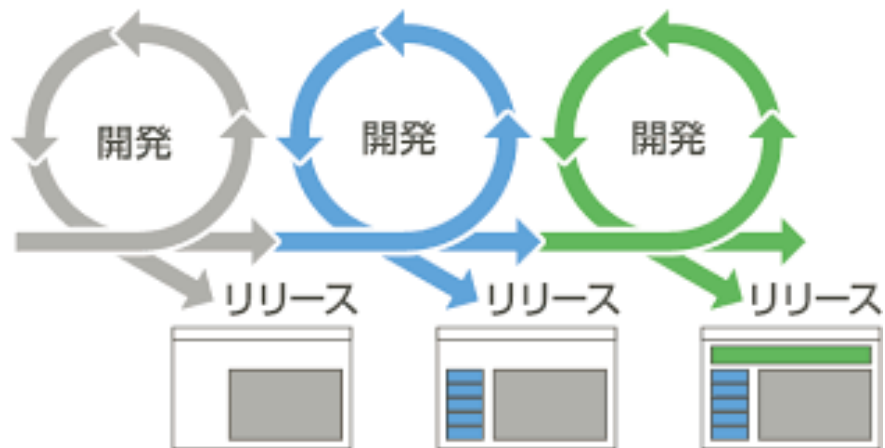
## Wモデルのメリット

設計段階でテスト工数が生じるかわりに、全体の工数、特にテスト最終段階での工数を削減でき、結果として開発費の抑制が実現できる。

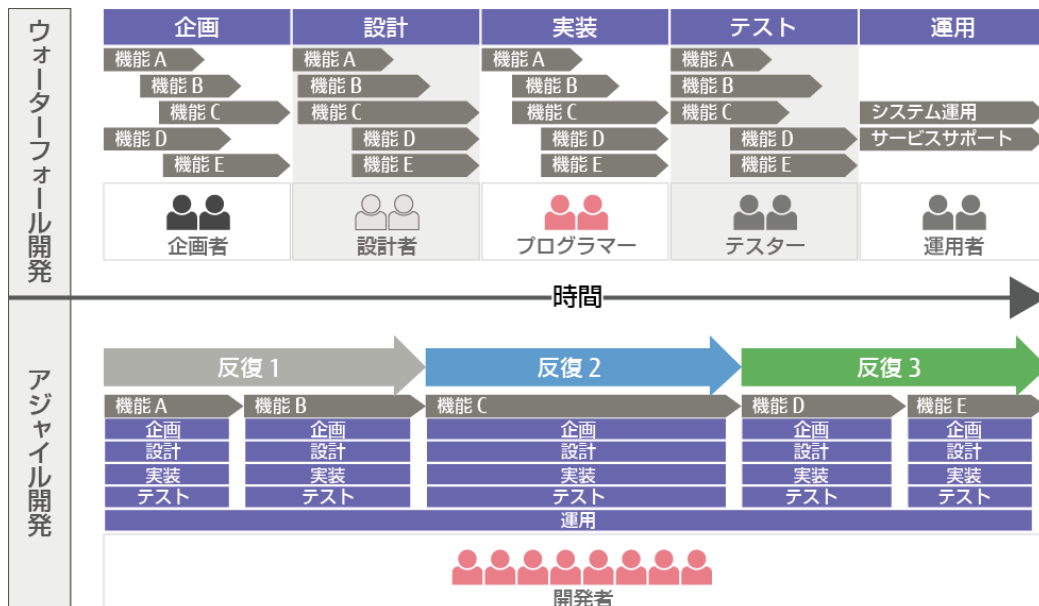
要件定義段階や設計段階でなければ改善が困難な、設計に起因する品質向上が期待できる。

## アジャイル開発

アジャイル開発は、1週間から1か月の反復期間を設け、その反復ごとに機能の追加を継続する「反復増加型」の開発プロセスによって実現されます。



## ウォーターフォール開発とアジャイル開発





## ウォーターフォール開発とアジャイル開発の特徴

比較の観点	ウォーターフォール開発	アジャイル開発
ビジネスサイド／顧客の関与	<ul style="list-style-type: none"> <li>・開発開始前に要求を決定</li> <li>・開発終了後に受入の判定</li> </ul>	<ul style="list-style-type: none"> <li>・開発期間中に要求の変更や追加</li> <li>・開発期間中に受入の判定</li> </ul>
要求変更への対応	<ul style="list-style-type: none"> <li>・開発前の要求確定が原則</li> </ul>	<ul style="list-style-type: none"> <li>・開発期間中の変更を受入れる (反復ごとに開発対象を決定するため)</li> </ul>
提供スピード	<ul style="list-style-type: none"> <li>・全工程の完了後に提供</li> </ul>	<ul style="list-style-type: none"> <li>・速い (反復ごとに提供)</li> </ul>
ドキュメント	<ul style="list-style-type: none"> <li>・各工程の生産物として必須</li> <li>・後工程への唯一の情報伝達手段</li> <li>・工程完了のエビデンスとして必須</li> </ul>	<ul style="list-style-type: none"> <li>・適切な情報伝達手段であれば作成 (保守用途、リリースノートなど)</li> </ul>
テスト	<ul style="list-style-type: none"> <li>・テスト工程でのみ実施</li> </ul>	<ul style="list-style-type: none"> <li>・実装やビルドごと頻繁にテスト</li> </ul>
開発者	<ul style="list-style-type: none"> <li>・工程ごとの専任担当者</li> </ul>	<ul style="list-style-type: none"> <li>・すべての開発作業を担当する多能工</li> </ul>
開発者の教育／採用	<ul style="list-style-type: none"> <li>・専任技術者のため教育が容易</li> <li>・同じく採用も容易</li> </ul>	<ul style="list-style-type: none"> <li>・反復開発で実践的に学習</li> <li>・全開発作業担当のため多能工化が促進</li> </ul>
プロセス改善	<ul style="list-style-type: none"> <li>・次プロジェクトに経験を活かす</li> </ul>	<ul style="list-style-type: none"> <li>・反復ごとにプロセス改善が可能</li> </ul>

## スクラムの特徴

反復増加型のソフトウェア開発プロジェクトを管理するためのフレームワークです。

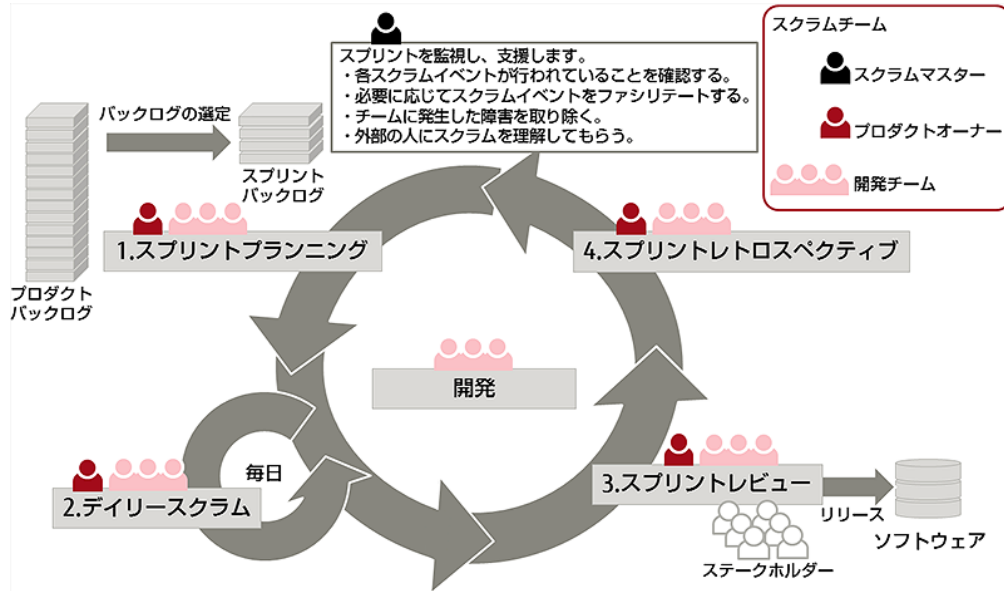
開発技術を必要とする活動はありません。

反復期間内に開発する機能は、反復ごとに決定されます。

反復期間中、チームは外部（顧客やビジネスサイド）からの干渉を全く受けません。

プロジェクト管理の権限はチームに委譲され、開発手法や利用技術もチームが決定します。

## スクラムのプロセス



## エクストリームプログラミングの特徴

XP:eXtreme Programing

開発に関わるすべての人が互いに尊重し、働きかけ、協力し合うことが重視されています。

状況の変化を歓迎し、受け入れ、それにすばやく追従することができます。

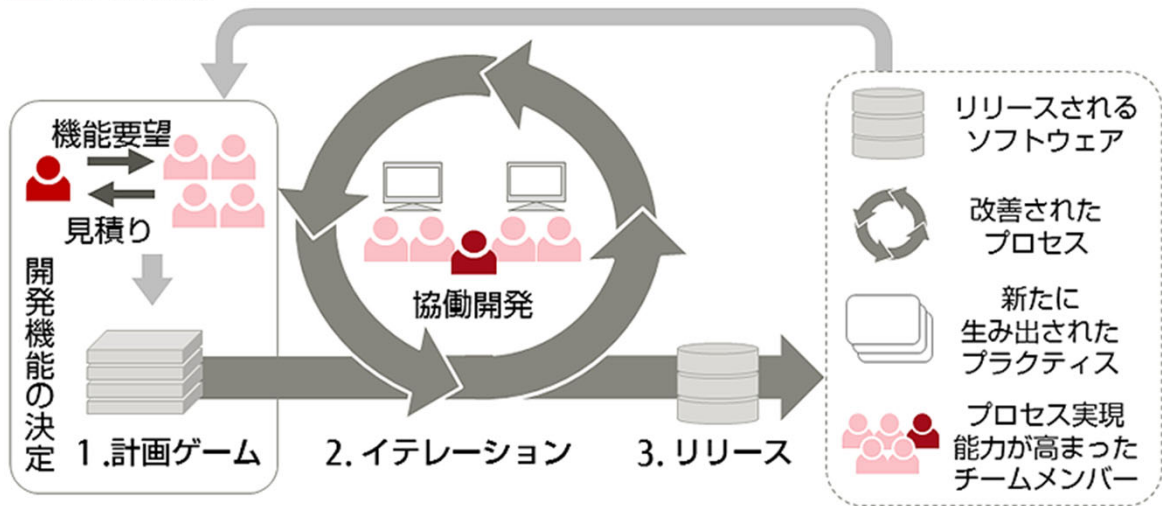
顧客はチームメンバーの一員です。オンサイト顧客と呼ばれチームと常に同席し、技術者と協働してソフトウェアを開発します。

開発プロセスはチームにより新たにつくられ、よりよいものに改良され、合わなくなったものは捨て去られます。

開発技術を必要とする活動が多くあります。

## エクストリームプログラミングのプロセス

👤 オンサイト顧客  
👥 開発技術者



## スクラムとXPの特徴

	スクラム	XP
実態	反復型開発をうまく回すための <b>仕組み</b> (フレームワーク)	よいものをつくるための <b>知恵の集まり</b> (パタン・ランゲージ)
チームとビジネスサイド の関係	スプリント期間中は没交渉 ・スプリントの前に要求確定 ・スプリント後に確認レビュー	全期間に渡って深く関与 ・要求の決定、変更、抹消 ・受入テストの作成実施 ・開発技術者と常に同席
プロセスの特徴	フレームワークの実行に開発技術は不要 ・外部干渉を遮断したブレイクスルー指向 ・ブレイクスルーを利用したプロダクトアウト指向	プラクティスの実行に開発技術が必要 ・プラクティス自体はマネジメント指向 ・顧客との緊密な関係はマーケットイン指向
適した領域	ブレイクスルーが必要な新製品の開発	継続的な成長を望む寿命が長い組織
チームに適した人材	能力の高い開発技術者たち	ふつうの開発技術者たち 能力が高ければ高くてもよい
人材とチームの能力	早期のチームビルディングが可能 個々の高い能力がチームによって活かせる	プラクティスのほとんどは人材／チームの育成が 目的 人材／チームの成長は長期継続的

## CI/CD

CI/CD (継続的インテグレーション/継続的デリバリー) とは、アプリケーション開発のステージに自動化を取り入れて、顧客にアプリケーションを提供する頻度を高める手法です。CI/CD によって、統合およびテストのフェーズからデリバリー、デプロイメントに至る、アプリケーションのライフサイクル全体を通じて、継続的な自動化と継続的な監視が導入されます。これらの連結した作業習慣はまとめて「CI/CD パイプライン」と呼ばれ、アジャイルな方法で共同作業する開発チームと運用チームが実施します。

## 変更管理

### 変更管理の目的

- 変更により障害が起きた際に特定をするため履歴を残す
- 変更によるダウンタイムを最小限に抑える
- 変更対象外のシステムをコントロールする

### 変更プロセス

- 要求提起 (Submission)
- 計画 (Planning)
- 承認 (Approval)
- 構築 (Implementation)
- レビュー (Review)
- クローズ (Closure)



## 品質管理

### 品質管理

工程管理 作業手順標準化

品質検証

品質改善

## ポリシーのベストプラクティス

OS、ネットワーク、ホスト、コンテナ、アプリケーションおよびマイクロサービスレベルでポリシーを設定します。コンテナ、マイクロサービスおよびアプリケーションなどの数に関係なく、サービスレベルポリシーを使用してサービスを保護し、セキュリティを確保します。

マイクロサービスがプロパティファイルをロードできる開示された集中構成管理サーバを設置します。

OS、ネットワーキング、コンテナ、マイクロサービスコンポーネント、アプリケーションレベルでポリシー管理プラットフォームを採用します。使用するポリシー、それらのパラメータ、およびそれらを管理システムに適用および実装する方法を判断します。

詳細なポリシーを使用して、特定の深刻度のCVEの影響を受けるすべてのイメージなど、セキュリティ要件を満たさないイメージを検知します。

イメージのスキャンをして、ランタイムの適用と修正機能に対応させます。

## ロギングのベストプラクティス

すべてのマイクロサービスリクエストの呼び出しに一意のIDをタグ付けして、いかなるエラーでも、エラーが発生したサーバやコンテナとマイクロサービス(アプリケーション)の呼び出しや応答を、破棄された後であっても追跡できるようにします。

エラー応答を一意のIDでコードにし、それら(コンテナおよびマイクロサービスのエラー)をより簡単にグループ化および分析できるようにします。

コンテナとマイクロサービスのログデータは、標準形式(JSONなど)で構成します。

選択されたすべての標準形式のログフィールドを検索可能にします。

集計、分析、レポートの目的でメッセージのUTC(協定世界時)時間を記録します。

重要な情報の欠落を避けるために、記録を少なくするのではなく、より多くのデータを記録するようにしますが、コンテナとマイクロサービスのデータ量が原因で過剰な要求が発生する場合、開発者とオペレータは、削減を図ったり、オフラインストレージと分析ツールで作業できるように準備をする必要があります。

## ロギングのベストプラクティス

専用のログ配布コンテナを介して、ログを流れるデータのストリームとして調査します。

専用のログ配布コンテナを介して、すべてのログを一元管理された場所に転送し、コンテナとマイクロサービスのアクティビティのレポートと監視を容易にします。

コンテナのマイクロサービスが利用できない場合（コンテナが破棄された場合など）や、ストレージスペースのリソースの可用性のため、ログをホストの外部に保存します。

目的とするツール（高速、大量のデータの処理、可視化とAIの使用）のコンテナとマイクロサービスを採用して、ログアクティビティを保存、集約、およびレポートします。

## 監視システムとアラートのベストプラクティス

ホスト、コンテナ、マイクロサービスレベルで、エンドツーエンドの継続的に可視性の高い環境を構築します。  
create、run、kill、launch、syscall、seccompなどのコンテナやマイクロサービスコマンドを制限および監視します。  
ホスト、インフラストラクチャ、マイクロサービスレベルで主要な指標を特定して監視します。  
サービスレジストリを使用して、実行時に利用可能なマイクロサービスの場所を保持します。  
クライアント側やサーバ側の開示情報を使用して、使用可能なマイクロサービスを見つけます。  
複数のコンテナやマイクロサービスが「動作している」という観点でクラスタを監視します。  
(複数のホスト、インフラストラクチャ、マイクロサービスに)分散された脅威を関連付けます。

## 監視システムとアラートのベストプラクティス

コンテナやマイクロサービス情報の膨大で分散された性質に対応した、大規模で高速な相関分析を実装します。コンテナやマイクロサービスのふるまいをベースラインに比較して分析し、悪意のあるアクティビティを検知します。AI(予測分析)を使用したコンテナやマイクロサービスの要求量の分析により、ふるまい、パターン、分類を関連付けます。

解釈しやすく、すべての主要な指標を含むダッシュボードを実装します。

- a. ダッシュボードの指標からホスト、インフラストラクチャ、マイクロサービスのログにドリルダウンする機能を備えます。
- b. ホスト、インフラストラクチャ、マイクロサービスのログからダッシュボードの指標にドリルアップする機能を備えます。

DevSecOpsによってマイクロサービスチームメンバーに割り当てられたアラートに応じ、適切な実行責任者に直ちにアラートを送信します。

解決の方向性を含めた明確で実行可能なアラートを作成します。

# リスク対策

## リスク要因分析



## リスク

構築の前提となる現行の仕様情報の把握が必要  
対策

## リスク

コスト高い

対策

## リスク

開発工程になると再構築と同等となり統制が必要

対策

# 作業自動化ツール

## 運用ツール

分類	機能	
言語変換		
バッチ	バッチジョブ実行	
オンライン	トランザクションマネージャ	
	アプリケーション実行基盤	
ファイル、データ系	データベース	
	文字コードの相互変換	
運用管理	統合コンソール	
	ジョブ制御	
帳票・文書管理	電子帳票	

## 開発ツール

分類	機能	
言語変換		
業務調査ツール		
テスト自動化ツール		

## ビジネスルール管理ツール

機能: ルールとして定義した業務上の判断をアプリケーションから呼び出して実行する仕組み  
ビジネスルール管理ツールの導入で保守性を向上すると共に、利用状況を分析し業務上の課題を抽出することで、移行後の継続的な業務改善を支援します。

BRMS(Business Rule Management System)ツール  
RED HAT Decision Manager

## 調査ツール

仕様書調査ツール

ソースコード調査ツール



## 言語変換ツール

それぞれの言語仕様から当社が独自に開発した言語変換ツールやオープン系言語の言語仕様・部品・共通サブルーチン・テンプレートなどを用い、スピーディーな機械変換を行います

特長

- マイグレーションフレームワークを適用し、高い機械変換率と移行品質を保持
- 直接オープン系プラットフォームへリニューアル可能
- 処理パターンと部品による構造化を意識した言語変換

テスト自動化

運用保守

# ケーススタディ

## 経験

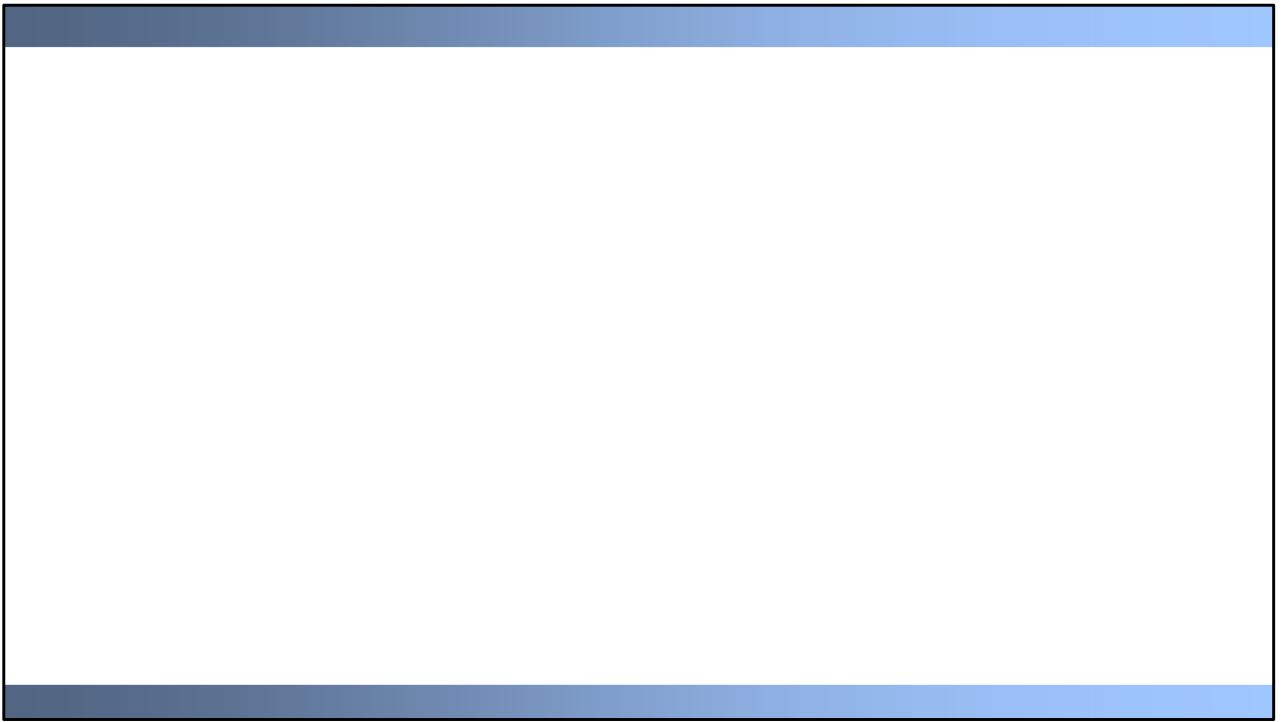
プログラム仕様の可視化と事実ベースの課題抽出  
システム構築作業の標準化  
業務ルールとプロセスの分離

## メインフレーム（汎用機）系言語からJavaへの言語変換

RPG、PL/1、COBOLなどのメインフレーム（汎用機）系言語の変換対象リソースに対し、当社独自の移行ツールを適用し、オープン系言語へ変換します。  
90%以上の機械変換率

## VB6.0、VB.NETからVisual Studio最新版への言語変換

古いOS(WindowsXPなど)で稼働しているVisual Basicを、最新OSで稼働するVisual Studioへ言語変換します。アップグレードウィザード(v6)と当社開発ツールを併用して移行することで、移行箇所のレポート出力も可能となり、保守性も確保できます。





## 参照資料

大宇宙ジャパン社内

資料共有(大宇宙社内アクセス可能)

[https://drive.weixin.qq.com/s?k=ANoAKQd\\_AAoZB0FD2Q](https://drive.weixin.qq.com/s?k=ANoAKQd_AAoZB0FD2Q)

## 参照資料

経済産業省

産業界におけるデジタルトランスフォーメーションの推進

URL:[https://www.meti.go.jp/policy/it\\_policy/dx/dx.html](https://www.meti.go.jp/policy/it_policy/dx/dx.html)

平成30年9月報告書『DXレポート～ITシステム「2025年の崖」の克服とDXの本格的な展開～』

平成30年12月『デジタルトランスフォーメーションを推進するためのガイドライン』(DX推進ガイドライン)

## 参照資料

経済産業省

デジタルトランスフォーメーションの加速に向けた研究会の中間報告書『DXレポート2（中間取りまとめ）』 URL:<https://www.meti.go.jp/press/2020/12/20201228004/20201228004.html>

DXレポート2（サマリー）（PDF形式：1,142KB）PDFファイル

DXレポート2（本文）（PDF形式：4,670KB）PDFファイル

DXレポート2（概要）（PDF形式：2,956KB）PDFファイル

対話に向けた検討ポイント集 第1章（PDF形式：4,106KB）PDFファイル

対話に向けた検討ポイント集 第2章（PDF形式：2,576KB）PDFファイル

対話に向けた検討ポイント集 第3章（PDF形式：2,916KB）PDFファイル

## 参照資料

独立行政法人 情報処理推進機構(IPA)

「システム再構築を成功に導くユーザガイド～ユーザとベンダで共有する再構築のリスクと対策～」(独立行政法人 情報処理推進機構(IPA)2017年発行)

## 参照資料

Cloud Security Alliance (CSA)

「安全なマイクロサービスアーキテクチャ実装のためのベストプラクティス」（2020 年 11 月 17 日）  
Cloud Security Alliance (CSA)が公開している「Best Practices in Implementing a Secure  
Microservices Architecture」の日本語訳です。

<https://www.cloudsecurityalliance.jp/site/wp-content/uploads/2020/11/best-practices-in-implementing-a-secure-microservices-architecture-J.pdf>

## 未着手

リスク要因チェックリスト  
資産管理リスト  
見積書  
業務要件  
工程計画表、参照P136

## R&D課題

セキュリティ設計とテスト  
ユーザー認証  
サービス間の認可