＜PMAJ例会議演資料＞　配付用

システムズエンジニアリングって？

～日本のSEのためのシステム開発アプローチの再考～

2019年2月22日

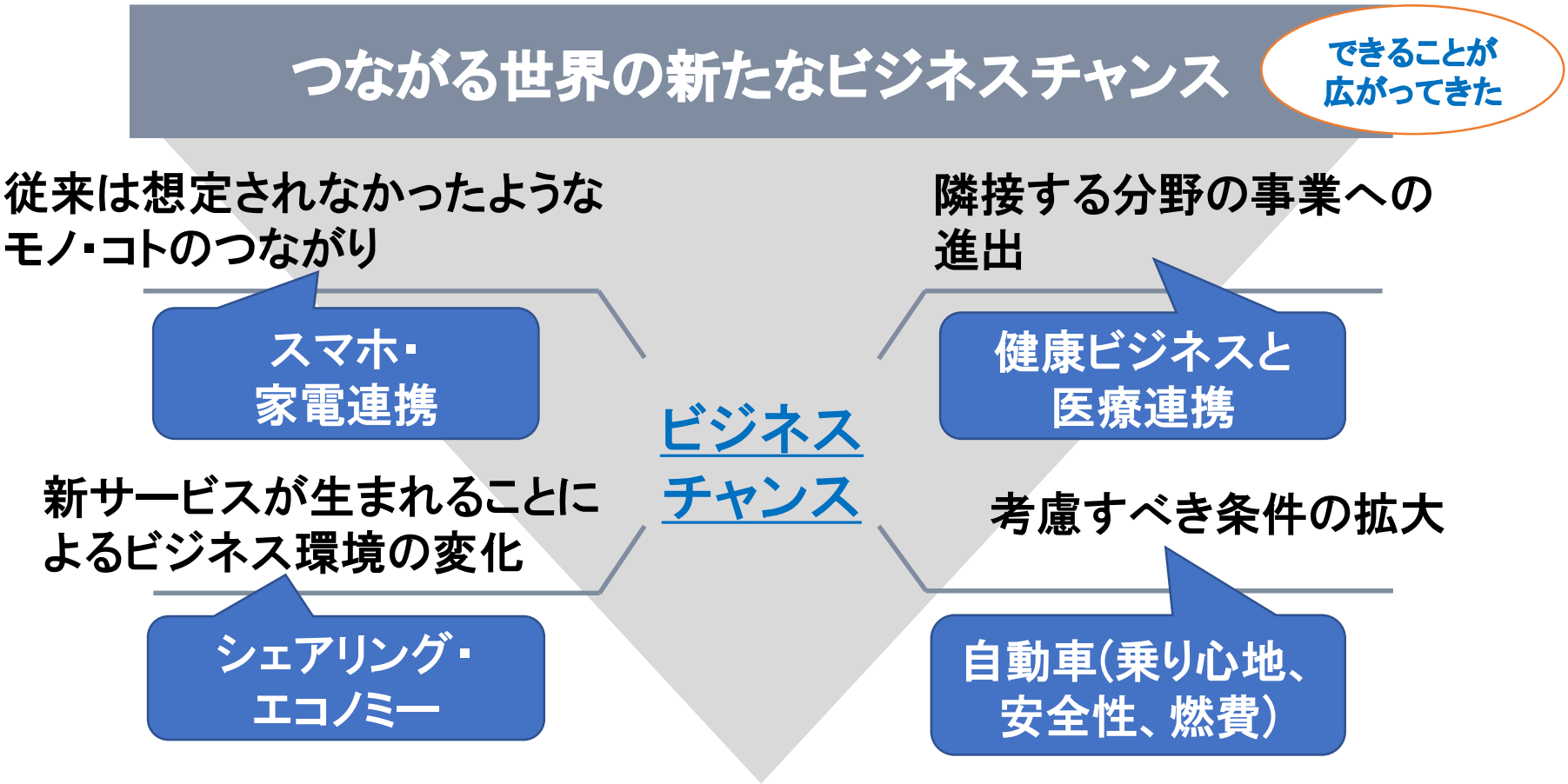
独立行政法人情報処理推進機構
社会基盤センター　イノベーション推進部
エンジニアリングG
齊藤善治

目次

- システムズエンジニアリングの概要
  - ✓背景
  - ✓4つのポイント
  - ✓システムライフサイクルプロセス
  - ✓成功事例に学ぶ効能
  - ✓一般情報からみたSysE（ものづくり白書他）

- 日本のSE（システムエンジニア）に向けて

IoT時代のシステムの課題



IoT時代のシステムの課題

ビジネスチャンスの裏には経営リスクも！

従来は想定されなかったような
モノ・コトのつながり

- つながる相手への迷惑、相手からの迷惑

隣接する分野の事業への進出

- 単一分野でのビジネスルールが通用しない

想定リスク

新サービスが生まれることによるビジネス環境の変化

- 現ビジネス領域の衰退

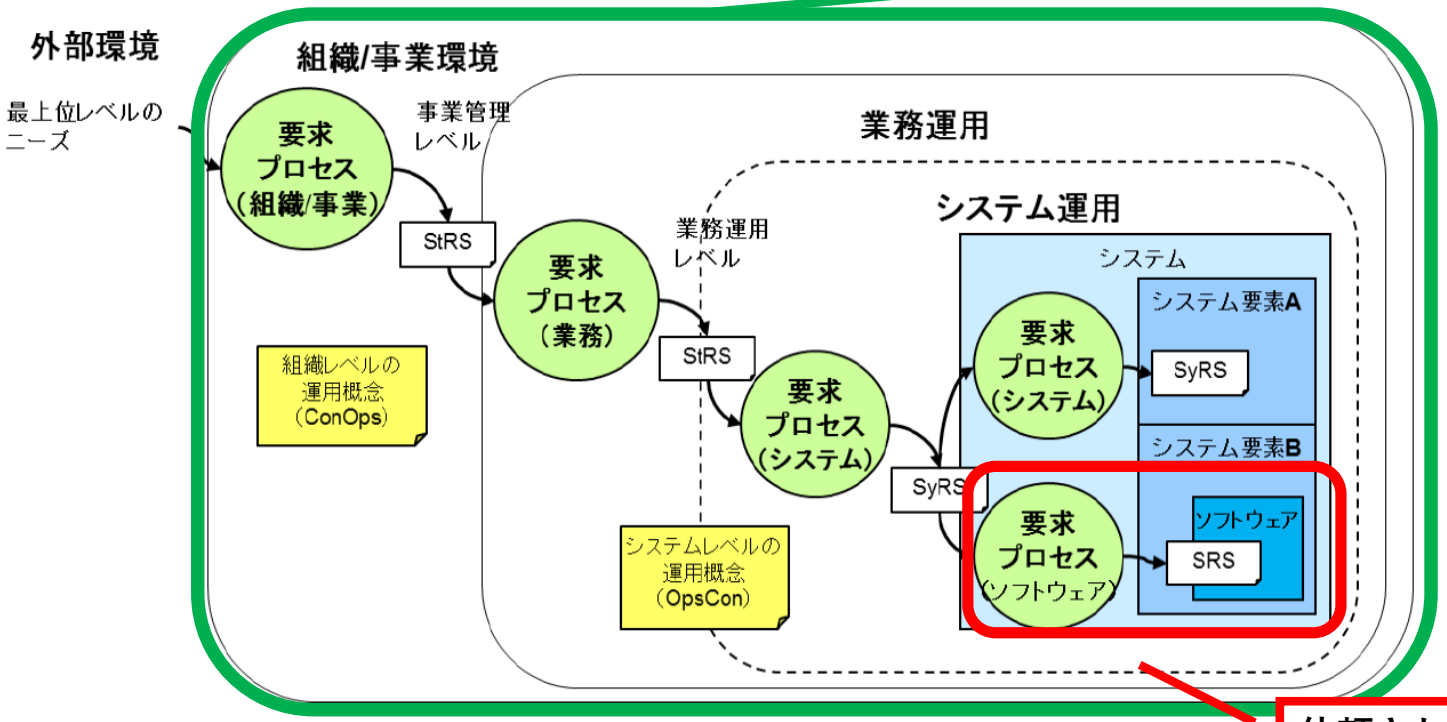
考慮すべき条件の拡大

- 考慮もれによる失敗（不備、遅延、事故）

新たなアプローチが必要

システムズエンジニアリングにおいて考える対象

役に立つシステムを作る



依頼されたソフトウェアを作る

StRS：ステークホルダ要求　SRS：ソフトウェア要求

SyRS：システム要求

ISO/IEC29148　JIS原案より

システムズエンジニアリングとは？

「システムを成功させるための複数の専門分野にまたがるアプローチと手段である」

JCOSE(Japan Council on Systems Engineering)

ここでいう「システム」は、コンピュータシステムにとどまらず、機械、電気機器、人間系（操作者）、環境など広い意味を表す。

システム：　構造を持った要素の集合。
全体として、要素にはない振る舞いや意味を発揮する。

航空・宇宙領域で確立した企画・開発のアプローチを汎用的に体系化したもの　⇒　欧米を中心に発展

参考文献：　INCOSE Systems Engineering Handbook:
A Guide for System Life Cycle Processes and Activities, 4th Edition

システムズエンジニアリング（SysE）はどのような場合に役立つのか？

多様な人の関わり

多様な利害関係者や専門家を含んだプロジェクトを実施しようとしている

付加価値の高いサービス

これまで単品の製品を開発し、一定の成功は収めてきたが、その製品を含めたより付加価値の高い総合サービスを実現したい

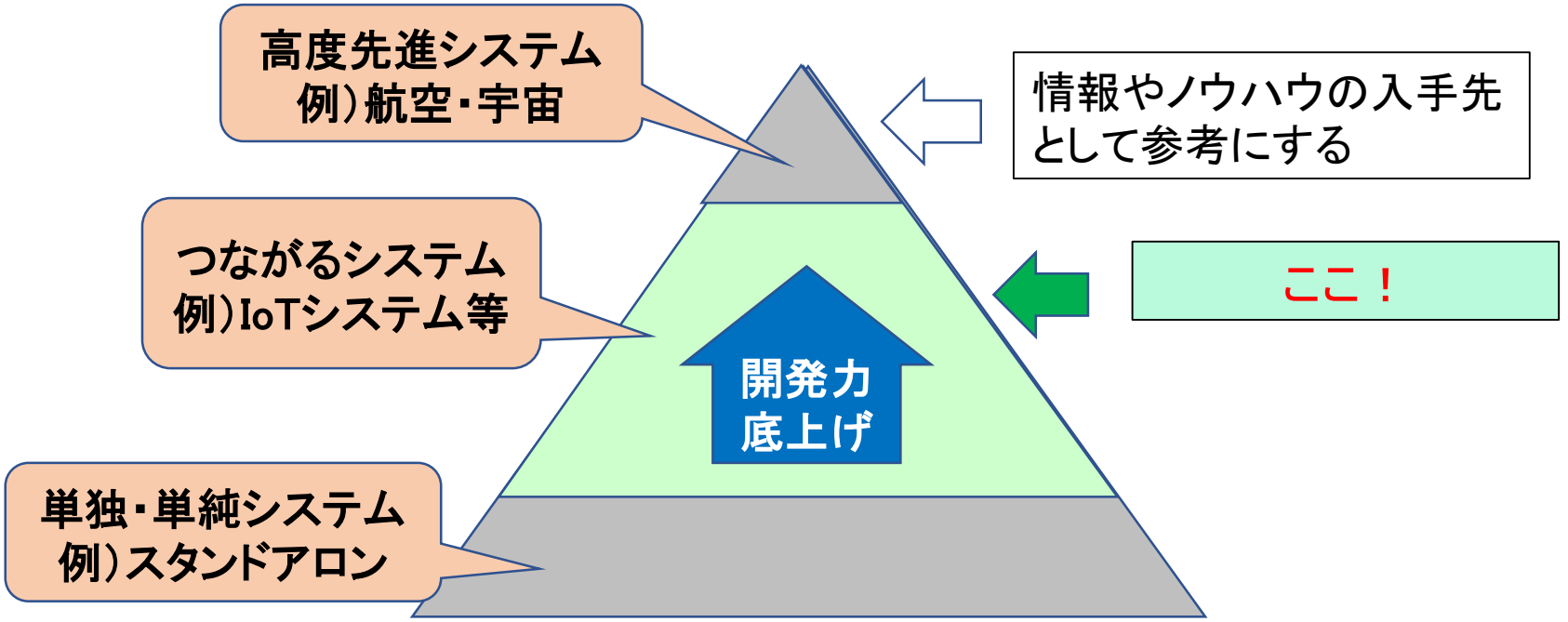
一段高い視点からの分析

要件が決まればきっちり作る自信はあるが、自らの技術・製品を取り巻く環境を一段高い視点から分析しなければならなくなった

→ 「IoT時代／デジタル変革時代」のシステム開発アプローチの要件

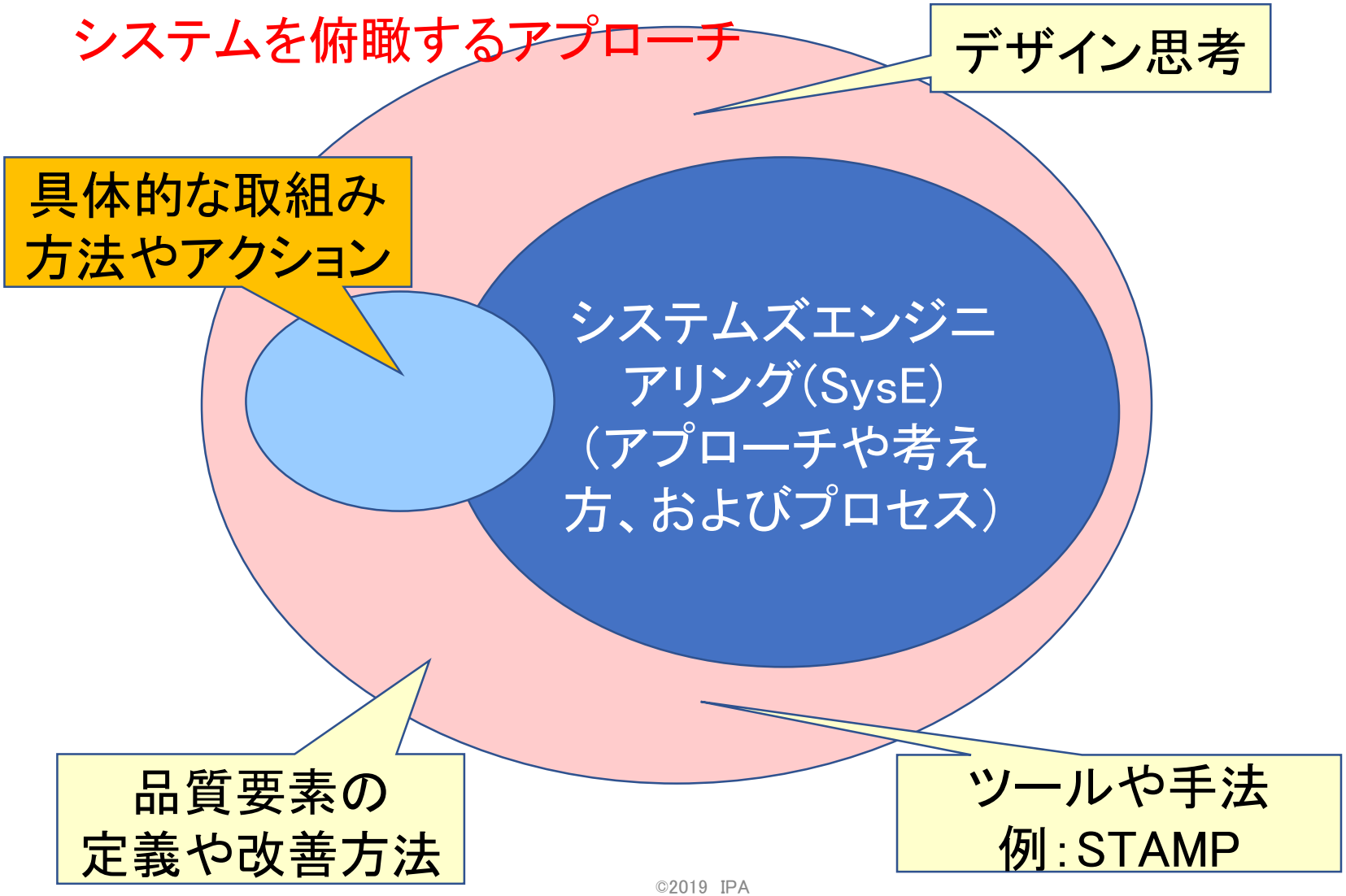
SysEの展開ターゲット

- 航空宇宙分野で確立されたシステムズエンジニアリングを、近年のIoTシステムの様につながって多様化する一般システムや製品の開発に適用し、開発力の底上げに寄与する

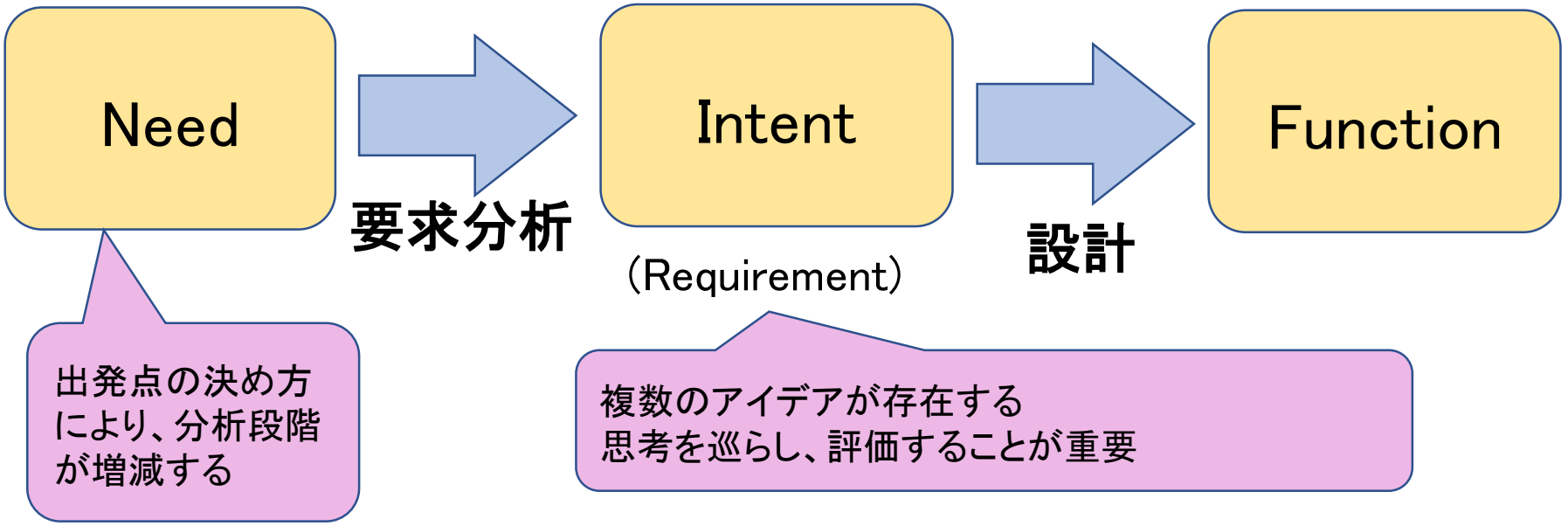


SysEのメリット
- ◆ 協創：　多様の専門家、利害関係者による新たな製品・サービスの創造
- ◆ 考慮範囲：　複雑な「つながり」から生じる広範なリスク・問題への対応

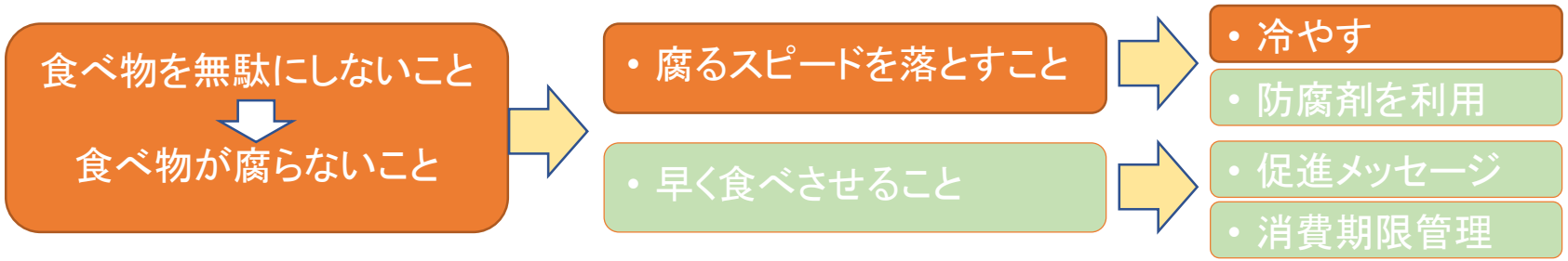
システムを俯瞰するアプローチとSysE



システムズエンジニアリングの思考過程



【冷蔵庫設計に至る例】



システムズエンジニアリングの4つのポイント

参考:「経営者のためのシステムズエンジニアリング導入の薦め」(IPA/SEC)

①目的指向と全体俯瞰

- 解決策を考える前に本来の目的を明確にし、常に目的を意識しながら考える。
- 視点と視野を変えながら全体を俯瞰して捉える。

②多様な専門分野を統合

- 多様な専門分野(技術、事業、領域、環境、文化、社会など)の知見を統合する。

③抽象化・モデル化

- 対象を抽象化・モデル化することにより、多様な専門分野の関係者の共通理解、本質理解の促進を図る。

④反復による発見と進化

- 適切に再評価とフィードバックを反復して、新たな解決方法を発見し、段階的に明確化・進化させる。

4つのポイントの実践に向けて
（1）－1　目的指向を実践するには

本来の目的の明確化

〇解決策に重点がいき、本来の目的を忘れがち
　⇒本来の目的意識の共有機会を持つこと
　（例）保育器：　赤ん坊の命を救うことが目的であり、
　　保育器を作ることが目的ではない。

〇部分最適が必ずしも全体最適とは限らない
　⇒色々な立場から検証すること
　（自己中心的な考え方から脱却し、価値観の多様性を認識）

妥当性確認(Validation)の実施

〇システムはうまく作れたが、役に立たない
　⇒Verificationだけでなく、Validationも（V&V）

仕様通り
VS
現実に使える

注）ここに記載した事項は、「目的指向」の留意点であり、システムズエンジニアリングの進め方を示しているわけではありません。

システムズエンジニアリングの事例

参考:「経営者のためのシステムズエンジニアリング導入の薦め」(IPA/SEC)

目的指向と全体俯瞰

- 発展途上国の実情に合わせた保育器の開発

  乳児死亡数が年間400万人に達している途上国に向け、より多くの生命を救うべく、新生児向けの保育器の開発・普及を行った事例

  - 開発上の課題

    既存製品を使用したが環境、インフラ環境による故障多発や部品入手困難のための修理網の整備遅れの結果、普及に失敗

  - 対策

    - 製品の本来の目的に立ち戻った新たな製品企画
    - 抽象度を上げた分析による本質的な要件および実現策の検討

  - 効果

    途上国で入手できる自動車部品で新たに開発し、普及に成功

出典: SEBoK(Guide to the Systems Engineering Body of Knowledge)

４つのポイントの実践に向けて
（１）－２　全体俯瞰を実践するには

【全体俯瞰に向けた俯瞰軸】

空間軸

例えば、対象システム（製品）の空間的利用環境を全て洗い出す。

物（影響のある範囲、つながる相手、・・）、場所（国、寒冷地、交通網、・・）、法律の制約　等

意味軸

例えば、誰が何の目的で利用するかを洗い出す。

登場人物、各立場からの利用目的、利用条件、・・・

時間軸

例えば、対象システム（製品）のライフサイクル全般を考える。

開発時、出荷後の初期設定時、利用時、休止時、更改時、破棄時

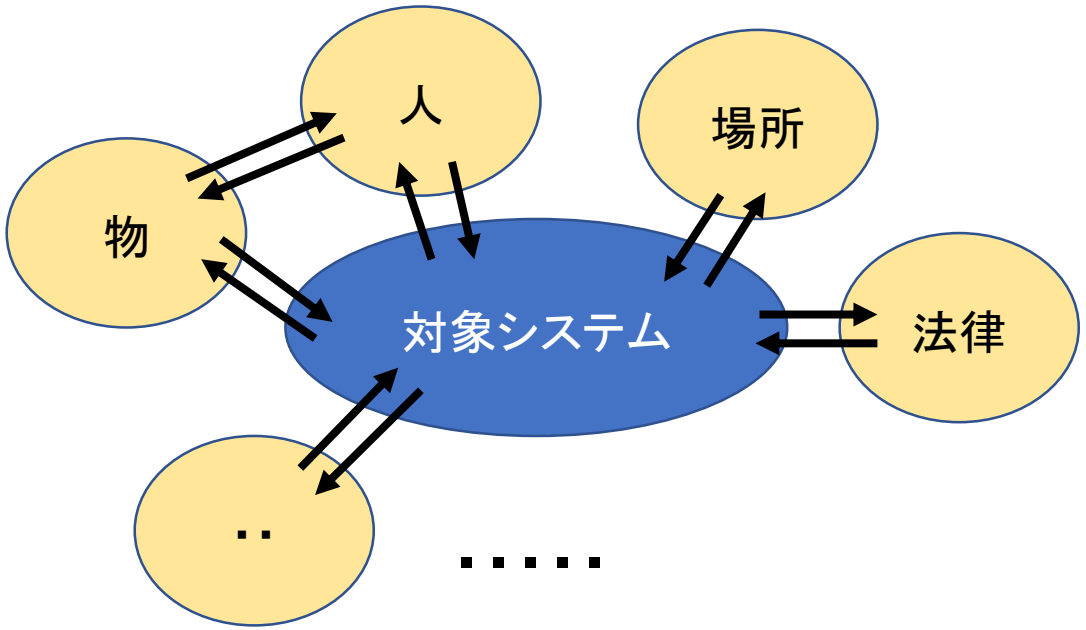
運用中に想定される環境変更も考慮

空間軸や意味軸を考えるには

コンテキスト図を書いてみよう！

対象システムとそれを取り囲む**環境・関係性**を洗い出す。

⇒考慮すべき事項への気づき



【関係性の例】

〇人の行う操作

〇人に表示する
メッセージ

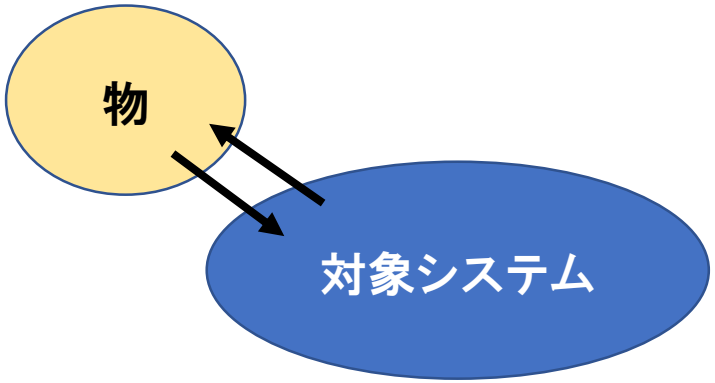
〇センサへのデータ
取得指示

〇センサからの
取得データ送信

例えば、つながる物に変化が生じたら

これまでつながっていた物は

スマホ、センサ、課金システム、
エンドユーザ端末、テレビ



つながる物が追加されたら

＋自動運転車、＋冷蔵庫

- 自動運転車が暴走したらどうしよう。
- 冷蔵庫が壊れたら何か対象システムに影響が？
- セキュリティも考えなきゃ。

多面的に考えを巡らして、気づきや考え漏れ防止に

認知バイアスに注意

4つのポイントの実践に向けて
（2）多様な専門分野の統合のためには

言葉・概念・常識の共有

〇分野毎の言葉・概念・常識は、通用しない（誤解は怖い）

⇒一般用語で話すことを心掛け、聞く側も少し違和感を感じたら、確認することを徹底

（例）「品質」という言葉は、業界によって異なる

調整役・組織・会議体・記述物が必要

〇体制やルールがなければ、統合は具現化されない

⇒全体調整にはルールや場所が必要

〇対立が発生してからでは、調整の場を作ることも難しい

⇒意識合わせは早い時期に

注）ここに記載した事項は、「多様な専門分野を統合」の留意点であり、システムズエンジニアリングの進め方を示しているわけではありません。

4つのポイントの実践に向けて
（3）抽象化・モデル化の実践に向けて

図や表で表す

⇒可視化は共通理解と以後の見直しに有効

〇図で表示すれば、概念的な捉え方の補助になるが、つい技術的な細かなことを書き込みがち（→理解されない、無視される）

⇒図の目的は抽象理解の補助なので、ポイントを絞った記述に

〇表は、整理軸の共通理解や網羅性確認に有用

⇒整理軸に一般性があるか、
軸の中に次元の違うものが含まれていないかをチェック要

重要要素と無視要素の認識合わせ

〇システムを取り巻く要素を全てモデルの中に取り込むのは、所詮不可能

⇒考慮する要素とそれに絞り込んだ（他を無視した）考え方（理由）を明記する（後で見直しがありうる）

注）ここに記載した事項は、「抽象化・モデル化」の留意点であり、システムズエンジニアリングの進め方を示しているわけではありません。

4つのポイントの実践に向けて
（4）反復による発見と進化をもたらすには

固定部分と見直し可能部分の整理

〇表面的な仕様見直しに終始するのでなく、反復の過程で骨組みを築き上げる

（例）基盤のアーキテクチャ

⇒固定部分と見直し可能部分を明示する

実験室データと実用データの差を認識

〇万全に作った気になっていても、実用に出すと色々な問題が噴出

（例）指紋認証は、実験室データでは十分な認識率を誇っていても、現場では、薄暗い環境、指紋が薄い人の存在、濡れた手等によって揺らぎが出る。

⇒現場実験の繰り返し実施とフィードバック

注）ここに記載した事項は、「反復による発見と進化」の留意点であり、システムズエンジニアリングの進め方を示しているわけではありません。

4つのポイントの実践に向けて
（4）反復による発見と進化をもたらすには

不確定要素を解明する仮説設定

〇できるところからやっても、先送りした問題は解決しない
⇒前進することで、判断材料を増やす
楽なところへ行くのではなく、見通しの利くところへ行く

ああ　アジャイルのことだね
とか
うんうん　PoC（概念実証⋆）　のアプローチだね
とか
それぞれ経験した例を思い浮かべてイメージしてください。

⋆効果効用の検証／技術的実現性の検証／　具体性の検証

システムライフサイクルプロセス

(理論的拠り所)

◆ISO/IEC/IEEE 15288:2015 Systems and software engineering – System life cycle processes

◆ISO/IEC /IEEE 12207:2017 Systems and software engineering – Software life cycle processes

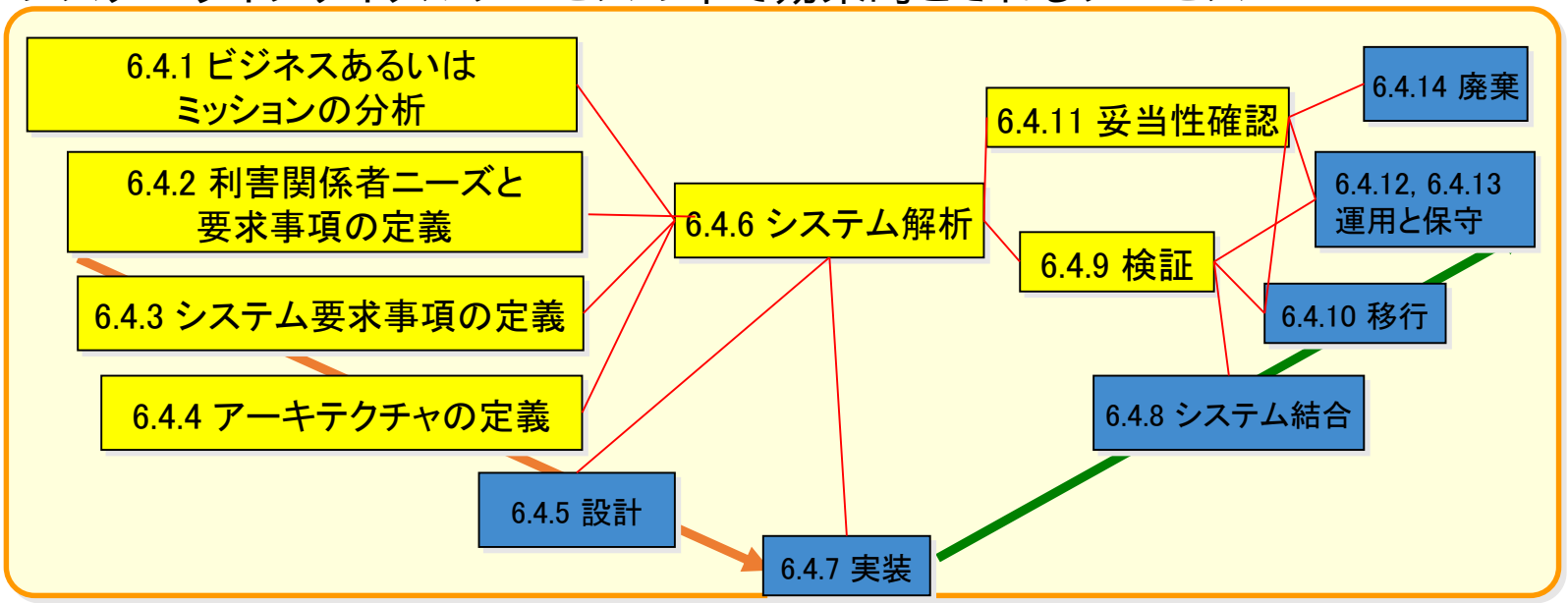
(参考資料)

◆共通フレーム2013　～経営者、業務部門とともに取組む「使える」システムの実現～　(IPA/SEC)

□Guide to the Systems Engineering Body of Knowledge (SEBoK)

http://sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK)

システムライフサイクルプロセスの中で効果的とされるプロセス



テクニカルプロセス概説

参照:「成功事例に学ぶシステムズエンジニアリング～IoT時代のシステム開発アプローチ～」

システムズエンジニアリングのプロセスは、システムライフサイクルプロセスの国際規格ISO/IEC/IEEE 15288　にて定義・解説されている。
（下記は、その項番を記載）

6．4．1　ビジネスあるいはミッションの分析プロセス

6．4．2　利害関係者ニーズと要求事項の定義プロセス

6．4．3　システム要求事項の定義プロセス

6．4．4　アーキテクチャの定義プロセス

6．4．5　設計定義プロセス

6．4．6　システム解析プロセス

6．4．7　実装プロセス

6．4．8　結合プロセス

6．4．9　検証プロセス

6．4．10　移行プロセス

6．4．11　妥当性確認プロセス

6．4．12　運用プロセス

6．4．13　保守プロセス

6．4．14　廃棄プロセス

部分について、以下で説明

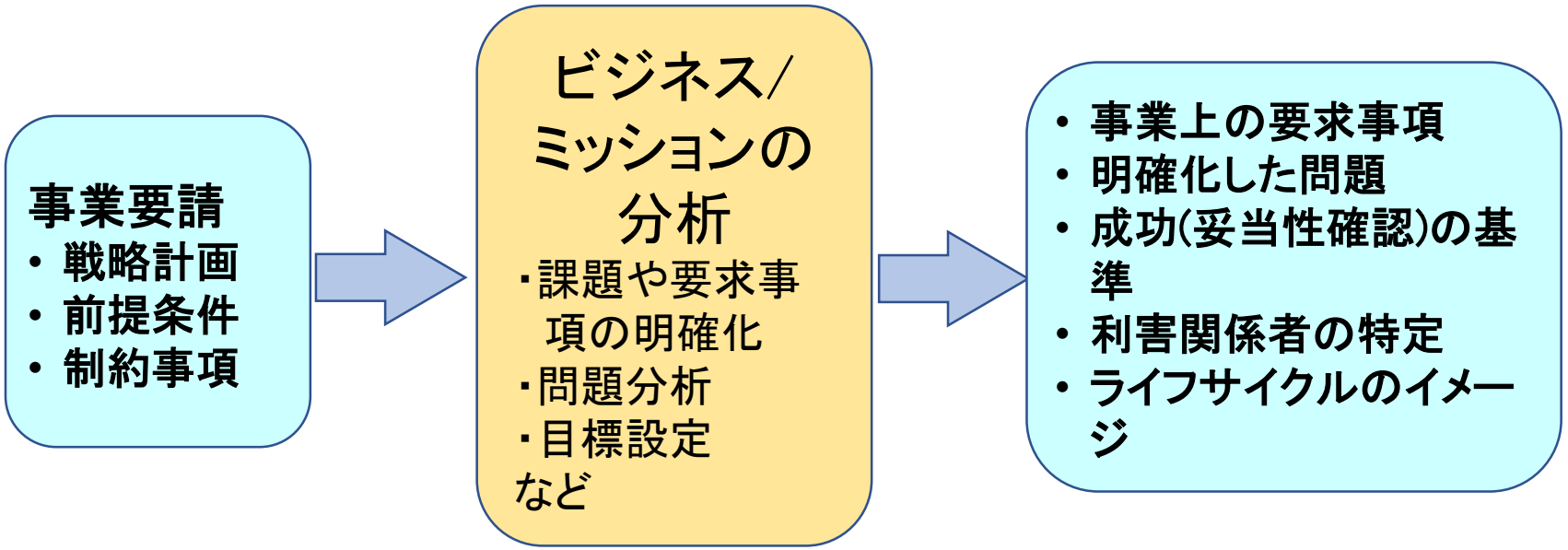
注）ISO/IEC/IEEE 15288の定義は解釈が難しいので、以下では平易な解釈文として記載

テクニカルプロセス概説

６．４．１　ビジネスあるいはミッションの分析

事業や任務（非営利活動の場合）の定義、目的などを明確にして、解決や達成のイメージを描く。

- 解決すべき問題を分析し、その本質を捉えて解決すべき課題を明確化する。
- 解決策としてどこまで考えるか、話を広げてみる



必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈が含まれています。

６．４．１　ビジネスあるいはミッションの分析

【具体例】

問題

自分の5歳の子供が野菜嫌いで食べない

ビジネス／ミッション分析

問題は
- 野菜嫌いか？
- 食べないことか？

まずは食べさせたい。
食べない原因は？
- 味が嫌い
- 親の料理がへた
- 見た目のすり込み

解決のイメージ

見た目の問題なら・・・
野菜だと気づかずに美味しく食べてくれるメニューを考案する。

問題である理由/解決することでの期待効果は?

野菜をたくさん食べれば、元気で賢い子になるはず

テクニカルプロセス概説

６．４．１　ビジネスあるいはミッションの分析

ビジネスまたはミッションの分析における留意点
（必ずしもISO/IEC 15288に記載されたものではなく、独自の解釈が含まれています）

出発点（本来の目的）の明確化

〇出発点を広めにとれば、思考は膨らむが、議論が発散

（例）冷蔵庫開発に至る思考の場合

- 食べ物を無駄にしない
- 食べ物を腐らせない
- 食べ物を冷やす

⇒どのレベルを出発点（目的）にするかという意識合わせが重要

テクニカルプロセス概説

６．４．２　利害関係者ニーズと要求事項の定義

利害関係者のニーズを把握し、コンテキスト等（周辺環境）を分析して、対象システムに対する要求事項を定義する。

ビジネス/ミッション分析結果

- 事業上の要求事項
- 明確化した問題
- 成功(妥当性確認)の基準
- 利害関係者の特定
- ライフサイクルのイメージ

利害関係者ニーズ/要求事項を定義する

- 利害関係者ニーズヒアリング
- コンテキスト図作成
- ニーズを要求事項に変換する
- 優先順位付け
- 移行/運用の方針検討

- ニーズ
- 要求事項
- 要求事項の優先順位
- 運用の考え方

必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈が含まれています。

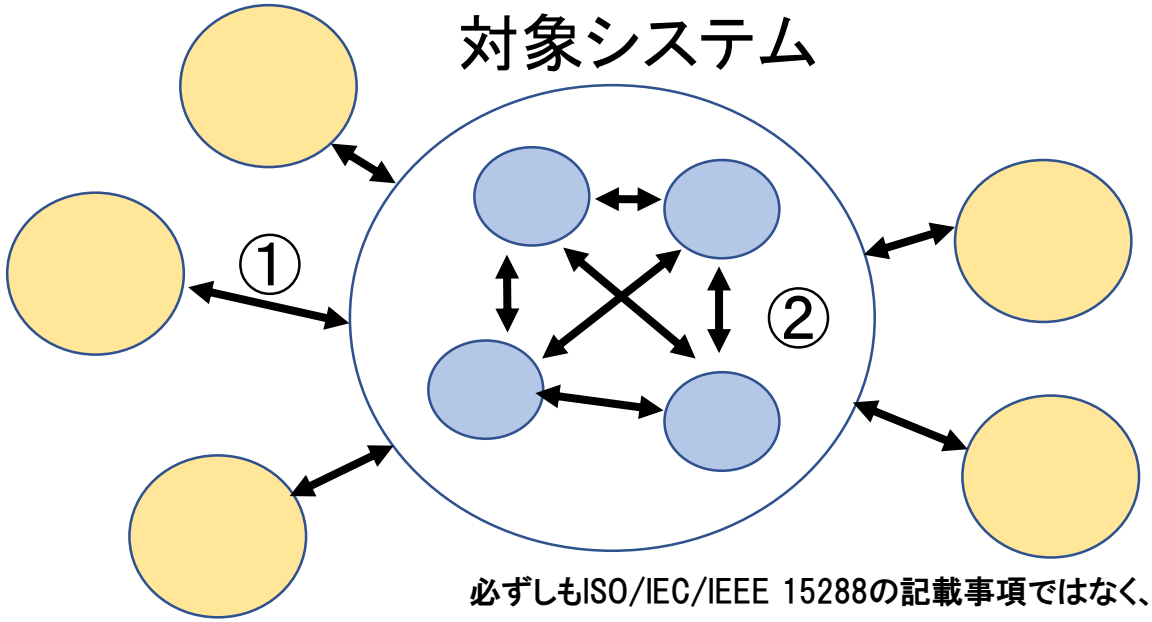
６．４．２　利害関係者ニーズと要求事項の定義

利害関係者ニーズと要求事項の定義における留意点

Black Box Requirement　であること

〇ついシステムの作りを意識してしまいがち

⇒システムの作りを意識するのは、システム要求事項を定義する段階とし、ここでは外部条件だけを考慮

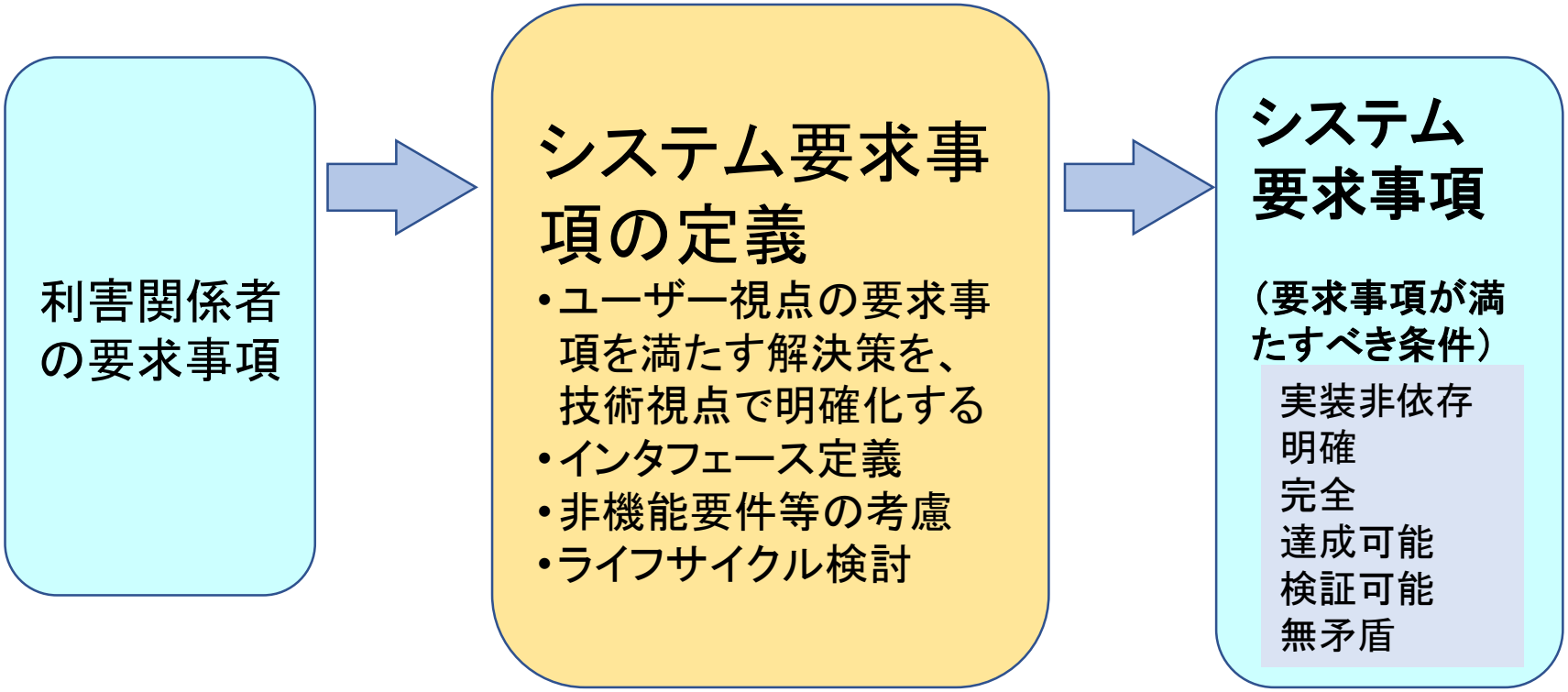


必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈を含む留意点です。

６．４．３　システム要求事項の定義

利害関係者の要求事項を実現するために、技術的に明確な表現で、システムがどう振る舞い、何を提供するのかを定義する。

利害関係者が陽に言わないことも、必要となる要求事項を追加する。
（業界の常識、法的要求事項、非機能要件、実装制約、運用条件など）



必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈が含まれています。

テクニカルプロセス概説

６．４．３　システム要求事項の定義

システム要求事項の定義における留意点

システム提供側のWILL（方針）を明確化

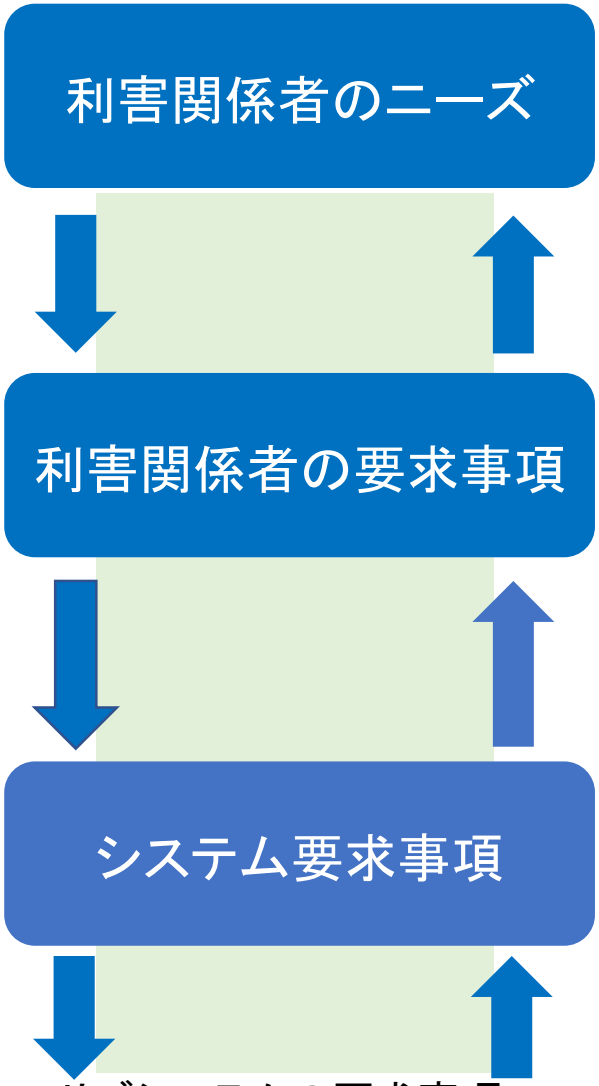
〇利害関係者ニーズ等の外部要件を全て実現することは不可能
⇒ニーズに応える事項の優先度付けを、システム提供側のポリシーとして示し、どこかに一線を引いて、実現する範囲を明確化

暗黙的な要件を明確化

〇利害関係者は、主に陽に特徴的な要件しか言ってくれない
⇒システムの専門家として、暗黙の要求事項や制約事項を導出
システムのライフサイクル全般の視点で考える（eg. 移行、運用、廃棄）
特に非機能要件の考慮が必要

必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈を含む留意点です。

補足：ニーズと要求事項の関係



- 利害関係者の言葉で、問題意識や期待が表現される
- 曖昧、不正確、達成判断不能な場合もある

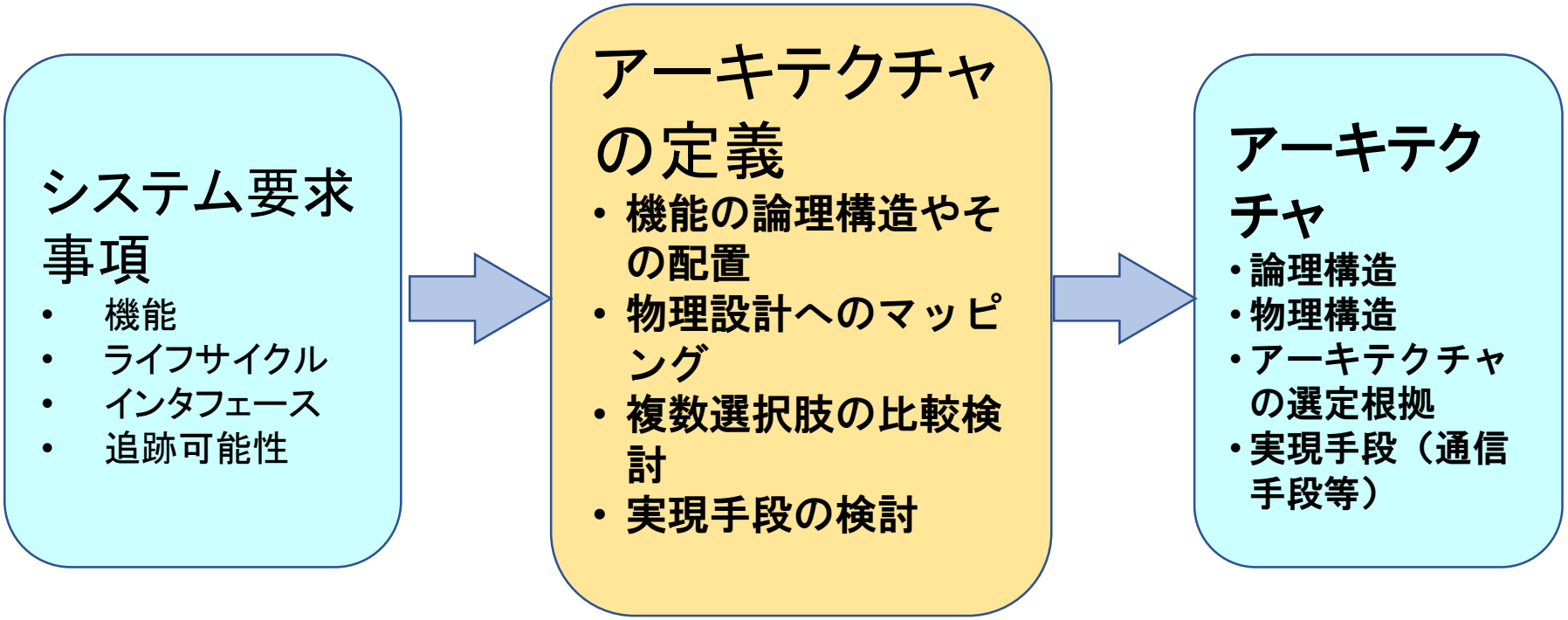
- 利害関係者のニーズをまとめて、プロジェクト実行の拠り所となる文書にまとめる
  - システムの機能やサービス
    （優先順位や品質レベルを含む）
  - コストやスケジュール、その他の制約事項
- 妥当性確認の基準になる

- 技術的用語でシステムを規定する
- 設計者に伝わらなければならない
- 検証の基準になる

テクニカルプロセス概説

６．４．４　アーキテクチャの定義

システム要求事項を実現するために、システムの基本的な構成要素や振る舞いを考え、具体的な機能や実現手段を構造的に示す。



必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈を含む留意点です。

テクニカルプロセス概説

６．４．４　アーキテクチャの定義

アーキテクチャの定義における留意点

機能と物理の分離

〇論理的な要求機能と物理的な実装を混乱して設計し、複雑化する。

⇒機能設計と物理設計を分ける

アーキテクチャの方針、考え方を大切に

〇アーキテクチャを逸脱すると、共有理解が困難

⇒論理階層やサブシステム階層を揃えた情報のやり取り、検討視点を。

アプリ同士、ミドル同士、OSレベル同士　ネットワーク論理階層

実現手段の比較検討

〇特定の考え方に囚われると目的を達成する有効な手段を見落とす

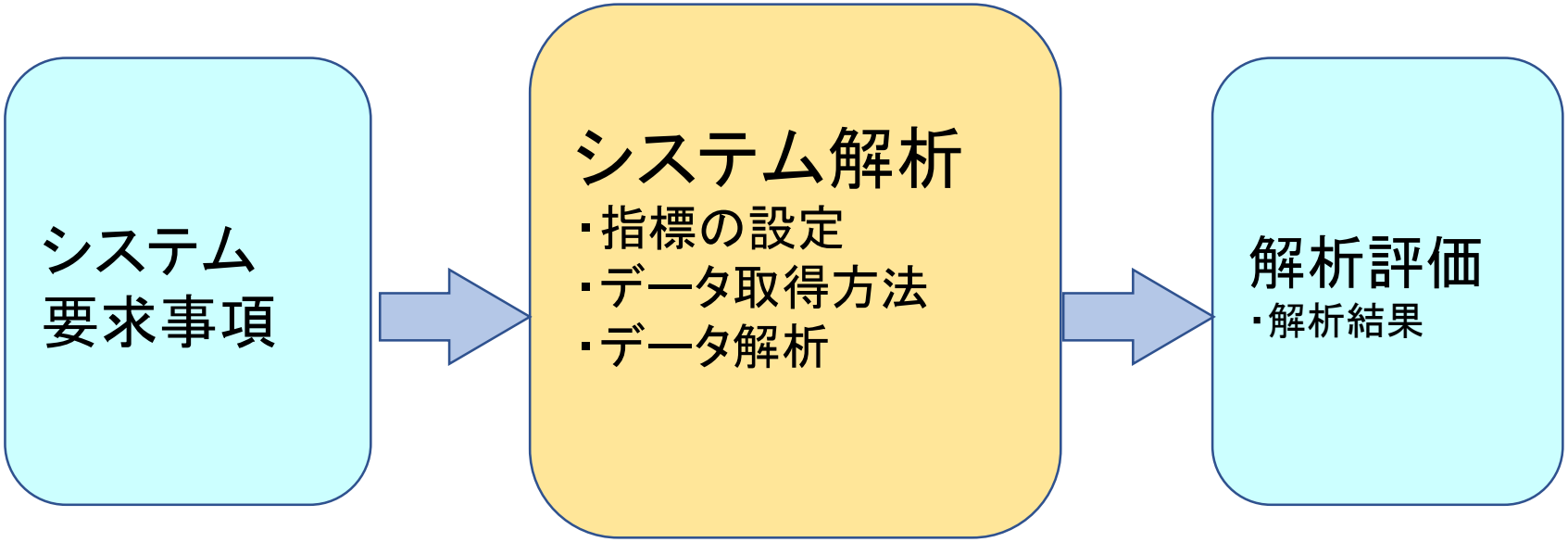
⇒柔軟な発想で多数の選択肢を抽出し、選定基準を明確にして、

比較検討する(Trade Study)

必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈を含む留意点です。

６．４．６　システム解析

システムの成果を評価するための指標、データ、解析を行う。
注）特定の工程にのみ存在する作業ではなく、開発のライフサイクル全般を通して実施するもの



システム解析の例：コスト見積、リスク分析、性能評価、実現可能性評価
品質要求事項を充足できるか推定

必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈を含む留意点です。

テクニカルプロセス概説

６．４．６　システム解析

システム解析における留意点

システムの技術的特性を指標定義、分析

〇設定した指標の重要性がわからなくなる
　⇒なぜ、どこまで、の根拠を追跡可能なように残す

〇一般に、機能、性能、信頼性、運用、拡張性等を主要な指標として設定するが、機能以外の設計条件が曖昧なケースが多い。
　⇒色々な角度から指標（〇〇性）を吟味し、大まかでもよいので、方針を定める。（次ページの例参照）

〇システムを特徴づける指標であるにも関わらず、出たとこ勝負のケースが多々ある。
　⇒システムを特徴づける指標（例えば、性能第一）は、必ず条件を明確化し、設計時にクリアできる見通しを得る。

必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈を含む留意点です。

例）大まかな信頼性条件

信頼性やセキュリティには万全というものは存在しない。
⇒方針（ポリシー）として対応レベルを設定する。

【ポリシー案】
〇顧客データだけは、強度に守る。ただし、コスト的に割り切る。（たとえば二重化まで。）
〇システムは最悪止まっても構わない。

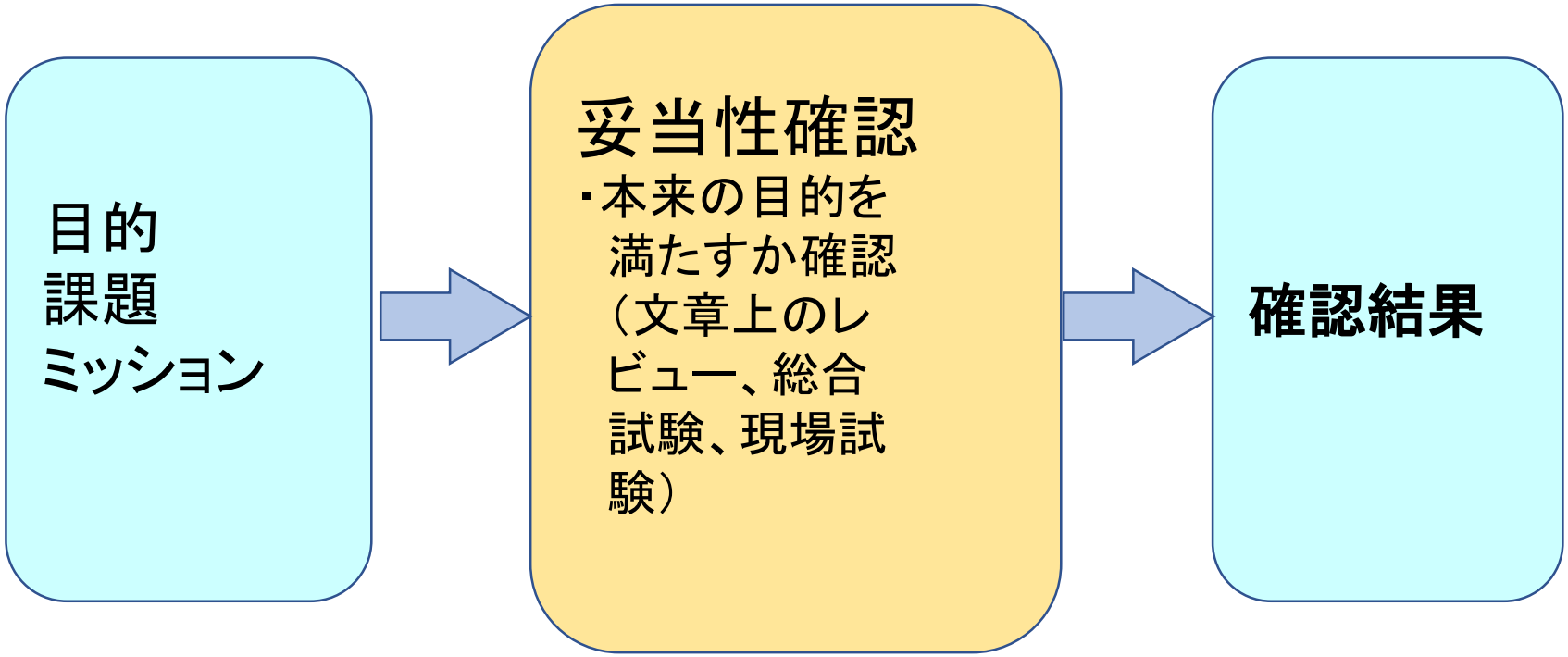
【理由】
・顧客データが破壊された時、システム以外に復旧手段がない。
・顧客データが盗まれると、会社責任が問われ、存続が危ぶまれる。
・システムが止まっても、今まで手仕事でやっていたことをシステムで自動化しただけなので、最悪でも運用でカバー出来る。

【具体策】
・顧客データのバックアップを取り、復旧可能とする。
・顧客データへのアクセス制御を行い、暗号化する。
・システムに何らかの重要異常を検知したら、システムを止めて、継続による重症化を防ぐ。

テクニカルプロセス概説

6．4．11　妥当性確認

システムがそもそもの目的を果たすことを確かめる。



必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈が含まれています。

テクニカルプロセス概説

６．４．１１　妥当性確認

妥当性確認における留意点

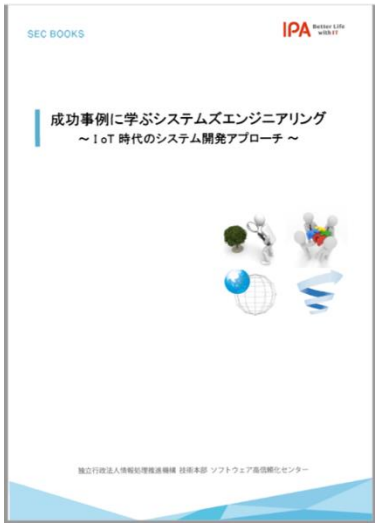
システムが出来上がってから確認するのではなく、随時確認が必要

〇システムはシステム要求事項を満たしたが、本来の目的を充たさないという事態が生じるのは、システム要求事項を明確化するまでの段階でミスがあったり、時間の流れとともに環境条件が変化することが要因

⇒妥当性確認（見直し）は、各開発工程のそれぞれで実施要

必ずしもISO/IEC/IEEE 15288の記載事項ではなく、独自の解釈を含む留意点です。

参考:「成功事例に学ぶシステムズエンジニアリング」の発刊（2018年3月）



「成功事例に学ぶシステムズエンジニアリング」

- 特徴:　複数の事例分析を通じて、システムズエンジニアリングのプロセスや重要ポイントを解説
- 想定読者: 製品/システム/サービスの企画・開発に取り組もうとするマネジメント層・リーダ・担当者

【入手方法】 https://www.ipa.go.jp/sec/publish/tn18-002.html

①書籍　・IPA直販、Amazon （500円）

②PDF　・IPAの公開ホームページよりダウンロード（無料）

「成功事例に学ぶシステムズエンジニアリング」掲載事例

	事例	企業
1	多様な関係者を巻き込み、ステークホルダのニーズと要求を　明確化し、全体を俯瞰し、段階的に集客イベントを支える情報共有基盤を開発、拡張して、**地域活性化**につなげた。	富士通総研
2	医療とITという複数の分野にまたがる複雑な問題に対して、抽象化・モデル化を活用した系統的なアプローチでセキュアな**電子お薬手帳**を実現した。	ソニー
3	**自動車エンジン**を全体最適の観点から設計し、個々の部品の物理設計に先行して機能開発することで、効率的に開発を進め、大幅な燃費向上を実現した。	マツダ
4	2世代先まで見通して、首都圏の高密度鉄道輸送を支える**デジタルATC**を実現した。移行や運用までも視野に入れて、試験時間帯の制約などの課題を克服した。	JR東日本
5	ビジネスシーンを俯瞰し、ビジネス分析およびステークホルダ要求分析を行って、**スキャナー**の新しいクラウド連携サービスを実現した。	キヤノン電子

システムズエンジニアリングの事例
Webスキャンシステム企画開発事例～概要～

■ 背景

- **競合の激化**
  スキャナー市場の伸び悩み
  新規参入メーカーと既存メーカーの間での競争
- **ペーパーレス化の動きが加速**
  デジタルネイティブのデータの増加
  紙を最初から使わないシステムの導入（例：インフォマートの電子取引）

■ 課題

- **他社との差別化**
- **新たなスキャナーの使い方の提案**

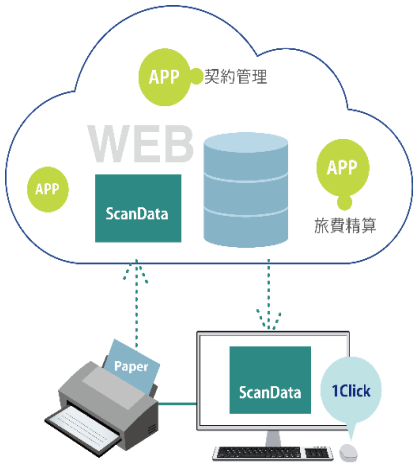
潜在的なニーズに対応し、マーケットの拡大を図る

Webスキャンシステム企画開発事例 ～アプローチ～

■ 対策の全体像

～ビジネスモデルの転換～

- **(旧)モデル**
  スキャナーを紙ドキュメントを電子化する機械として位置づけ
  - **業務は意識しない**
- **(新)モデル**
  スキャナーを使ってWebアプリケーションからワンクリックで使える電子化サービスを提供する
  - **業務やワークフローを意識**



"モノ"から"コト"へのビジネス転換の実現

ビジネスシーンにおける製品の立ち位置を見直し、新しい使い方の提案

タスクフォースを立ち上げて、ビジネスを俯瞰

あらたなビジネスの機会とステークホルダの発見

先進の米国でステークホルダへの
・ヒアリング
・要求の獲得
を実施

要求実現の為プロトタイピング開発

レビューによる妥当性確認とトレーサビリティの確保

事例による効能の説明
Webスキャンシステム企画開発事例　～対策詳細～

1　ビジネスの俯瞰

- 事業横断のタスクフォースの立ち上げ
- 視点を一段上げて、スキャナーが使われるビジネスシーンを想定して検討
- ビジネスシーンでの対象と想定する業務にASP事業者が提供するクラウド上のサービスを活用しているケースが多くみられることに着目
  - あらたなステークホルダの発見

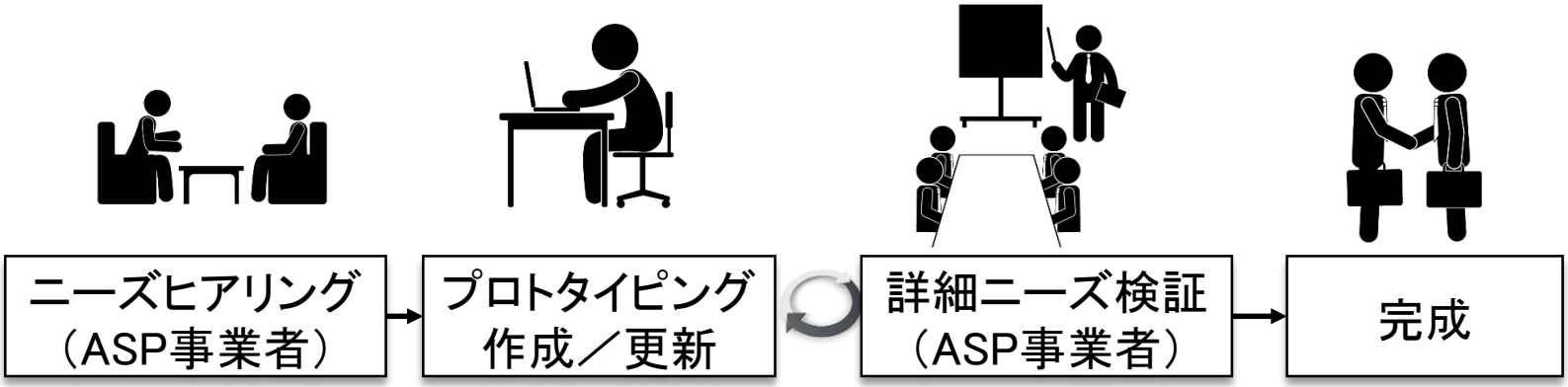
市場規模が大きくかつASP事業の先進である米国での調査を実施

【システムズエンジニアリングのポイント：目的指向と全体俯瞰】
【関連プロセス：ビジネスあるいはミッションの分析】

Webスキャンシステム企画開発事例
～対策詳細～

2 プロトタイピングによる反復

- ASP事業者から聴取した内容をベースにプロトタイプを作成
- プレゼンテーション・評価により詳細なニーズを発掘
  プロトタイプモデルに反映するというルーチンを繰り返し、合意形成
  - 早い段階から齟齬を解消し信頼を獲得

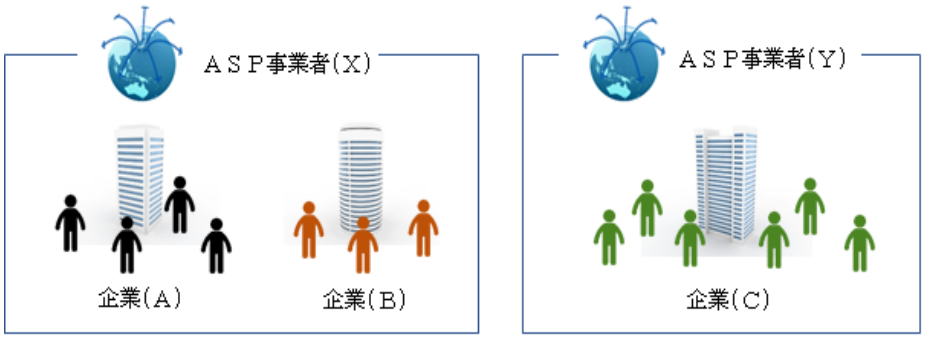


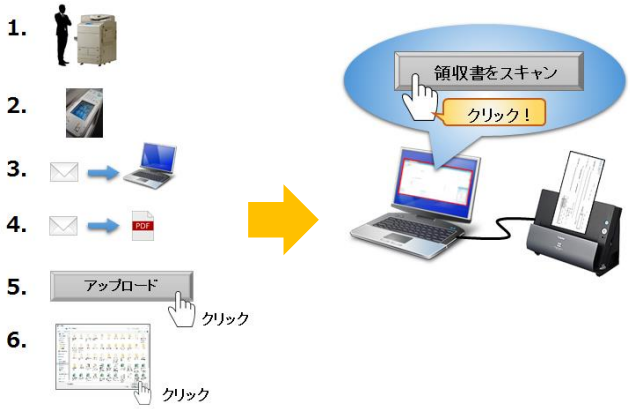
【システムズエンジニアリングのポイント：反復による発見と進化】
【関連プロセス：利害関係者ニーズと要求事項の定義】

Webスキャンシステム企画開発事例
～対策詳細～

3 ユーザーニーズを踏まえた機能提供

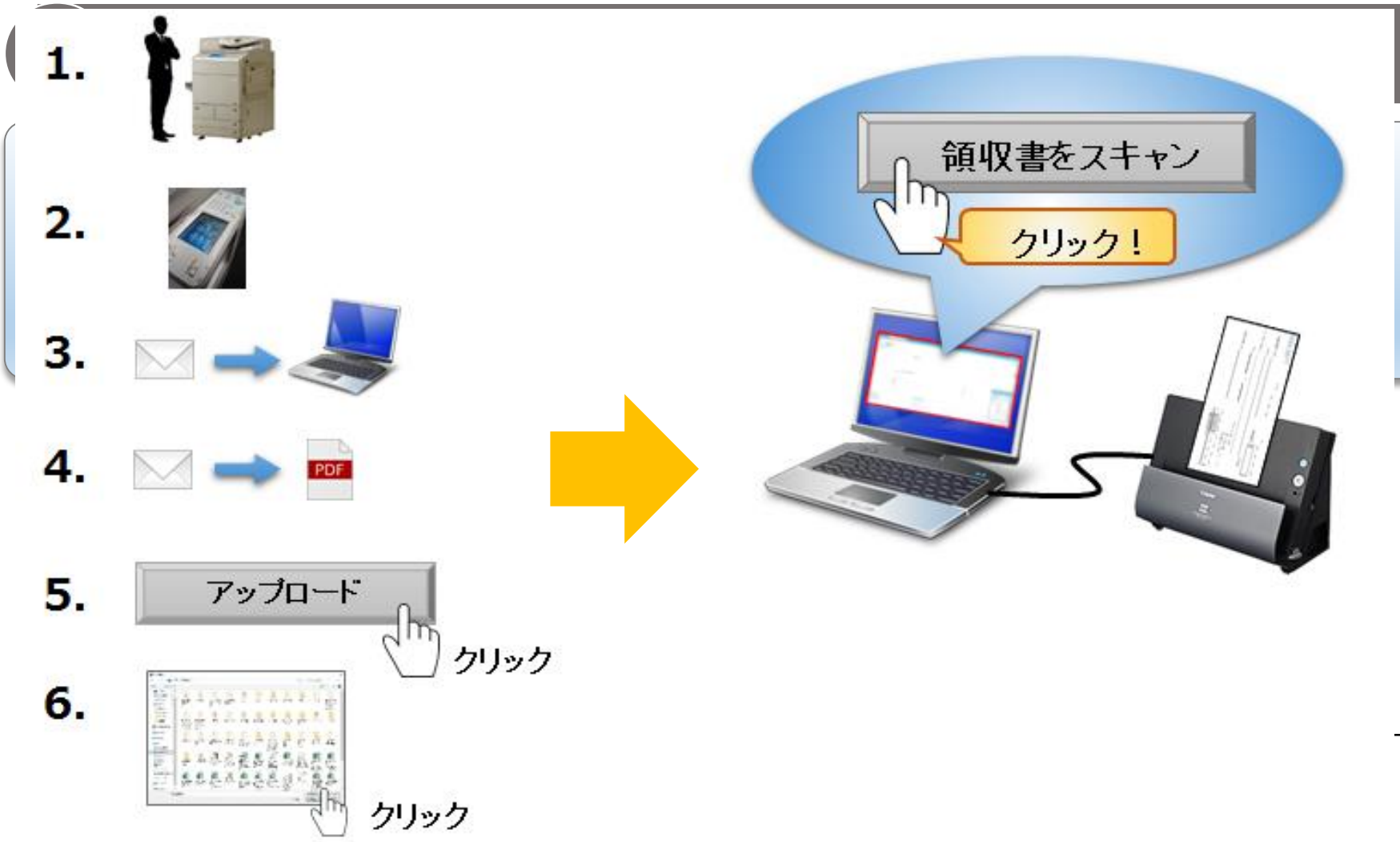
- エンドユーザーニーズを定義するにあたり、
  ASP事業者の背後には不特定多数の企業・ユーザーが存在することを考慮
- エンドユーザーのニーズの中で最大の関心事項が「コストの削減」
  作業レベルに分解して、効果を確認
  - 作業工数を削減することでユーザニーズ（「運用コストの削減」）を実現





【システムズエンジニアリングのポイント：目的指向と全体俯瞰、抽象化・モデル化】
【関連プロセス：システム要求事項の定義】

Webスキャンシステム企画開発事例
～対策詳細～



Webスキャンシステム企画開発事例
～対策詳細～

4 セキュリティの作り込み

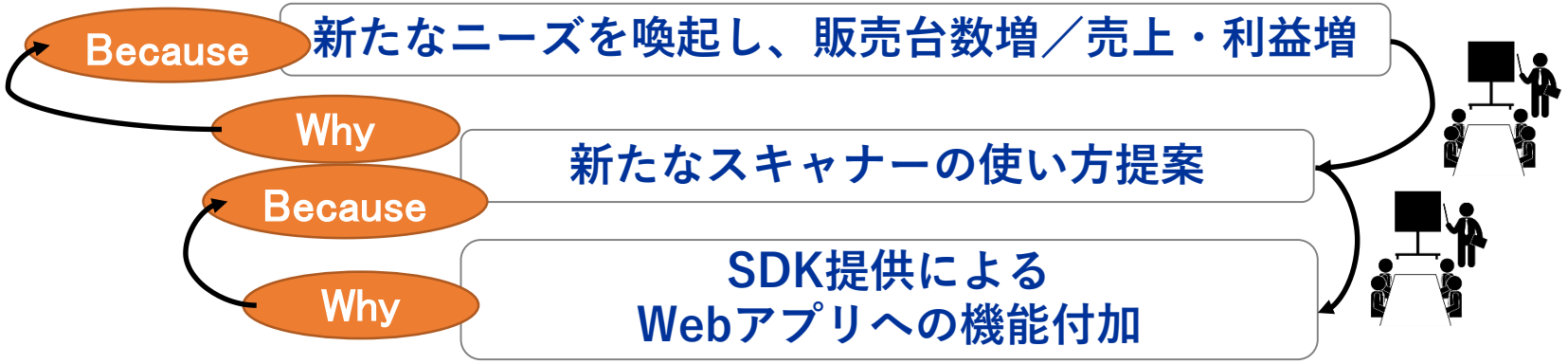
- アーキテクチャを検討する中で、ASP事業者にとって特に重要な関心事がセキュリティであることが判明
- 要求段階から検討してセキュリティの作り込み
  - 既存の部門に加えて、Web技術者やセキュリティ技術者などの知見を結集

【システムズエンジニアリングのポイント：目的指向と全体俯瞰、多様な専門分野を統合】
【関連プロセス：システム要求事項の定義】

Webスキャンシステム企画開発事例
～対策詳細～

5 トレーサビリティの確保

- ビジネス要求を実現するということを念頭に随時レビューを実施し、方向性やリスクの確認
- 決定事項には常に決定に至る理由を明記
  - 双方向（ビジネス要求⇔システム要求）のトレーサビリティを確保



【システムズエンジニアリングのポイント：目的指向と全体俯瞰】
【関連プロセス：妥当性確認】

Webスキャンシステム企画開発事例 ～教訓・まとめ～

- プロセスとシステムズエンジニアリングのポイントのまとめ

Web Scan SDK の企画開発

		目的指向と全体俯瞰	多様な専門分野を統合	抽象化・モデル化	反復による発見と進化	その他
プロセス	(6.4.1)【ビジネスあるいはミッションの分析】	市場規模が大きく且つASP事業の先進である米国での調査を敢行することでビジネスの骨格が固まった	他部門から決定権限者を招聘し、従来の発想にとらわれない視点でビジネスシーンを観察することでASP事業者という新しいステークホルダを見出すことができた			
	(6.4.2)【利害関係者ニーズと要求事項の定義】	ASP業者の最大の関心は、セキュリティであることが判明 ASPユーザーの最大の関心は、コスト削減であることが判明			ASP事業者から聴取した内容をベースにプロトタイプを作成し、プレゼンテーション・評価により詳細なニーズを発掘した	
	(6.4.3)【システム要求事項の定義】	ASP事業者の最大の関心は、セキュリティであることが判明した為、事前にリスク評価を実施した上でプロトタイプの製作に着手した	新たなマネジメントのもとに既存の開発部門加えて、Web技術者やセキュリティ技術者などの知見を集め合意形成活動を進めた	ユーザー業務を作業レベルに分解してBefore／Afterの形で整理した。作業自体を減らす又は負荷軽減することができるかを明らかにした		
	(6.4.4)【アーキテクチャの定義】					
	(6.4.6)【システム解析】					
	(6.4.9)【検証】					
	(6.4.11)【妥当性確認】	ビジネス要求を実現するということを念頭に随時レビューを実施				
	その他	決定事項には常に理由を明記した。 理由を明記することでビジネス要求側からシステム要求側へのトレース、 逆にシステム要求側からビジネス要求側へのトレースを確保した			プロトタイプ開発で合意形成を図った結果、両者の祖語を解消し信頼を獲得するに至った 米国のみに特化したニーズについてはフィルタリングすることでシンプルな仕組みとした	

参考　2018年版ものづくり白書で紹介

- 経済産業省

2018年版ものづくり白書

http://www.meti.go.jp/report/whitepaper/mono/2018/honbun_pdf/index.html

・システム思考、全体最適化の必要性に言及（第1部1章3節P168～P172）

・成功した事例を紹介した書籍として「成功事例に学ぶシステムズエンジニアリング」を紹介　（第1部1章3節P169）

参考　システム思考に関連する課題認識（公開情報より）

◆特定非営利活動法人横断型基幹科学技術研究団体連合

平成28年度　製造基盤技術実態等調査

（第4次産業革命における「知」のシステム化対応の実態調査）

報告書

http://www.trafst.jp/IRsys.html

◆ロボット革命イニシアティブ協議会（RRI）

『システム思考』ガイドブック（入門編）

https://www.jmfrri.gr.jp/document/library/913.html

その他　参考情報

◆ISO/IEC/IEEE 15288:2015 Systems and software engineering – System life cycle processes

◆ISO/IEC/IEEE 12207:2017 Systems and software engineering – Software life cycle processes

◆共通フレーム2013　～経営者、業務部門とともに取組む「使える」システムの実現～（IPA/SEC）

□Guide to the Systems Engineering Body of Knowledge (SEBoK)
http://sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK)

◆ システムズエンジニアリング導入実施の一事例　報告書（SEC　journal　第52号）
https://www.ipa.go.jp/files/000064317.pdf

◆ SEC iournal　第48号　（特集　システムズエンジニアリング　）
https://www.ipa.go.jp/sec/secjournal/048.html

◆ Harvard Business Review 2016年4月号
特集　デザイン思考の進化　　「はたして、論理は発想の敵なのか」（野矢茂樹）

(参考)妥当性確認

妥当性確認の上流工程での実践については、システムズエンジニアリングならではの知見ではなく、ソフトウェア開発の指針の中で従来より述べられている

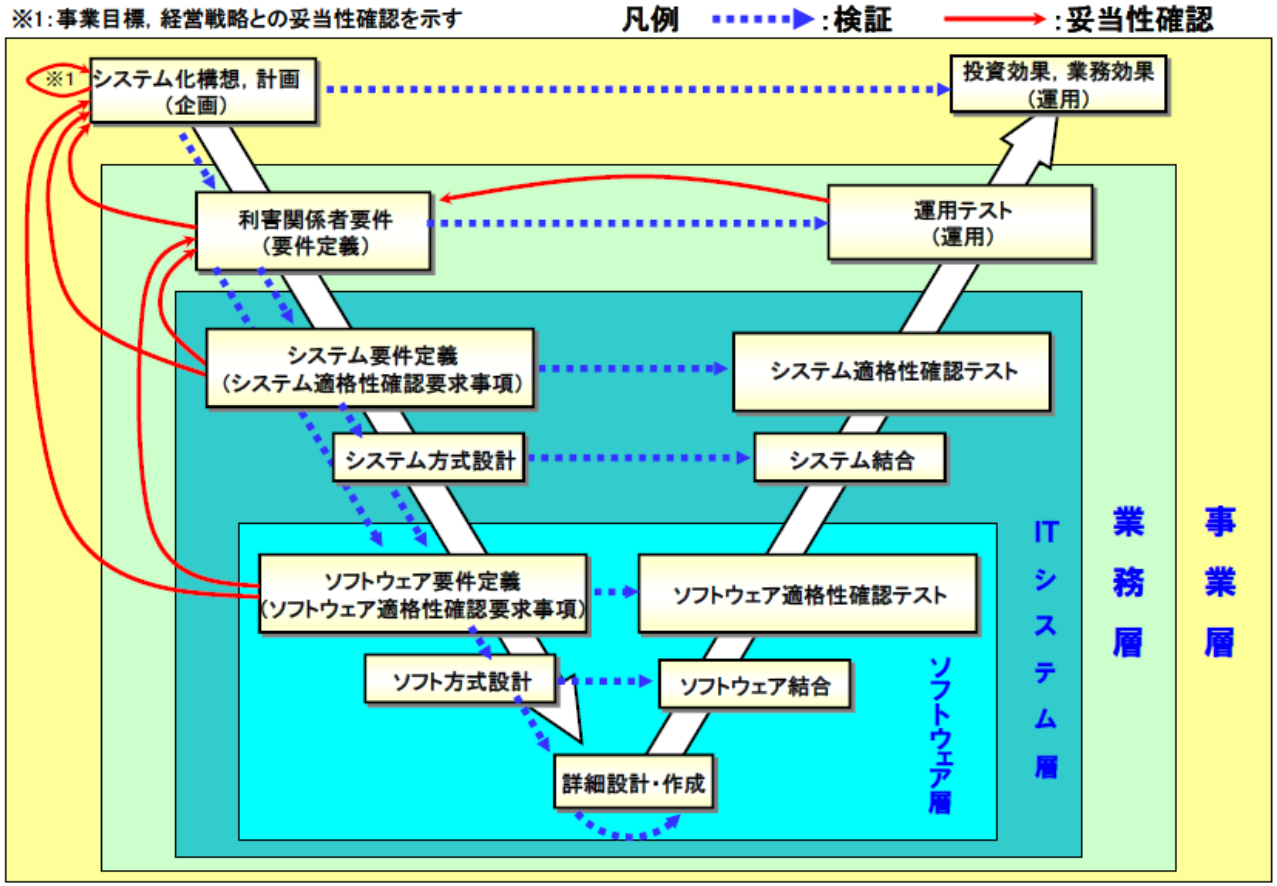


図4-xx 検証(Verification)と妥当性確認(Validation)の適用場面

※ 2013/3/7 IPA/SEC プロセス改善セミナー資料「共通フレーム2013の概要」(P58) を基に作成

- 日本のSE（システムエンジニア）に向けて

（SysEアプローチの実践例の紹介）
企業とのタイアップ　パイロットプロジェクト

システムズエンジニアリング導入実施の一事例　報告書　　https://www.ipa.go.jp/files/000064317.pdf

出典
・SECjournal第 52号「事例紹介：システムズエンジニアリングを活用したITSのセキュリティ機能設計の取り組み」（三菱重工業(株)原健太氏）（IPA）
・「システムズエンジニアリング導入の一事例　報告書」（2019.3.1）（IPA）

- **企業**

  三菱重工機械システム株式会社（MHI-MS社）

- **対象システム**

  ITS（Intelligent Transport Systems：高度道路交通システム）における向け***プロトタイプシステム***

- **パイロット活動の範囲**

  ITSプロトタイプ開発・実証実験プロジェクトの一部であるサーバ上のソフトウェア開発の「セキュリティ対応」設計活動の場に、***ライフサイクル***のレンジで捉えたシステムズエンジニアリングの考え方を取り入れる

イメージ図

パイロットプロジェクト実施の背景

「従来の知見（専門性やノウハウ）を強みとして活かしながら、直面する多様性や不確実性の課題に対処する新しいアプローチを必要としている」

従来の知見と、新たなシステムズエンジニアリングに関する知見を整合させる実践的知見が必要！

パイロットプロジェクト活動内容と成果

基本方針の確認

目的に即した全体俯瞰の重要性（目的指向と全体俯瞰の考え方）について意識合わせの上で推進）

	主な活動	主な結果
①	部門の開発標準を、システムライフサイクルプロセスの標準規格と比較	部門の開発標準の特徴を3点抽出 i. 「ビジネスあるいはミッションの分析」「利害関係者ニーズと要求事項の定義」の2プロセスに相当する内容は規定に含まれない。（→対応②） ii. 上流で考慮すべき範囲として、移行、稼働後の運用、保守への対応については規定されていない。（→対応③） iii. 上流の各工程での妥当性確認を行うことが明確に規定されていない。（→対応④）
②	ビジネス課題とシステム対策の関連を分析・整理	「プロジェクト要素分析シート」を作成して対象の分析を行い、**設計根拠**のドキュメントとして残した。
③	移行、運用・保守に関する課題管理	プロトタイピング段階の残課題を明文化し「システム構築手法説明書」としてドキュメント化して**課題管理**。
④	「設計工程での妥当性確認」の実施	要件検討時の資料に照らして、目的と設計内容の整合性のレビューを実施した。（課題発見12件）

活動を通して確認された知見

1. 開発標準を「システムライフサイクルプロセス」の標準規格と比較するアプローチは有効
   - 我流でのOからの検討を避けて、ソフトウェアに限らないシステムの視点が反映されたプロセスを活用できる
   - 自部門の(過去の)有識者の暗黙知が(よく考えられていることが)プロセスになぞらえて見えてくる副次効果があり、これを整理評価すると、新たな開発・運用や後進の育成に活用できる。
2. 上流工程の妥当性確認が有効
   - プロジェクト本来の目的とシステムの設計内容の整合性を設計工程から確認することで、問題の早期発見につながる。

プロトタイプらしくスピードをあげるためにいろいろ棚上げにすること → OK

何を棚上げしたのか理解しておくこと → 必要

企業とのタイアップ・パイロットプロジェクト（気づきのまとめ）

◆ソフトウェアの開発現場はそれぞれの製品・サービスの開発を通して培った開発標準を持っている。自プロジェクトに合わせてテーラリングして開発を行っている。

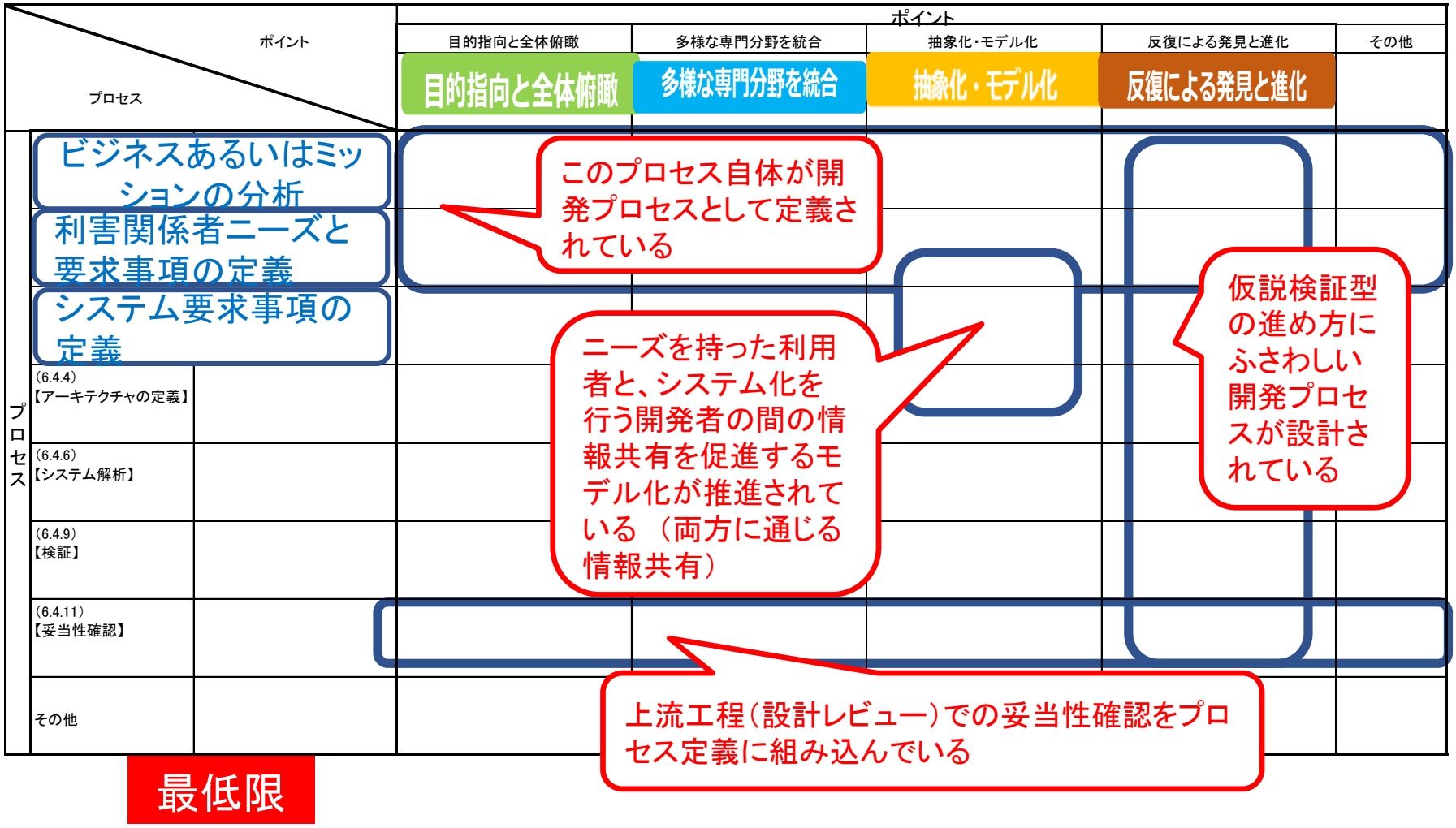
◆新しい取り組み(IoT、AIなど)に直面した人たちは、現在の開発標準が必ずしも今後に向けた”最適”ではないという現実に気づいている。

◆新たな環境には、多様な機器やステークホルダーなど、ソフトウェア以外の要素の影響が大きい。より広くシステムとして捉えた開発のために、システムライフサイクルの標準が役立つ。

◆システムズエンジニアリングとして体系化された膨大な知見の中から、開発分野の特性に応じた取り入れ方が必要であるが、システムライフサイクルプロセスは共通基盤として有効。

ポイント×プロセス対応表

■ 「システムズエンジニアリングのポイント軸」と「プロセス軸」との交点で開発標準を評価　→　「できている」と胸をはれる状態は、、、

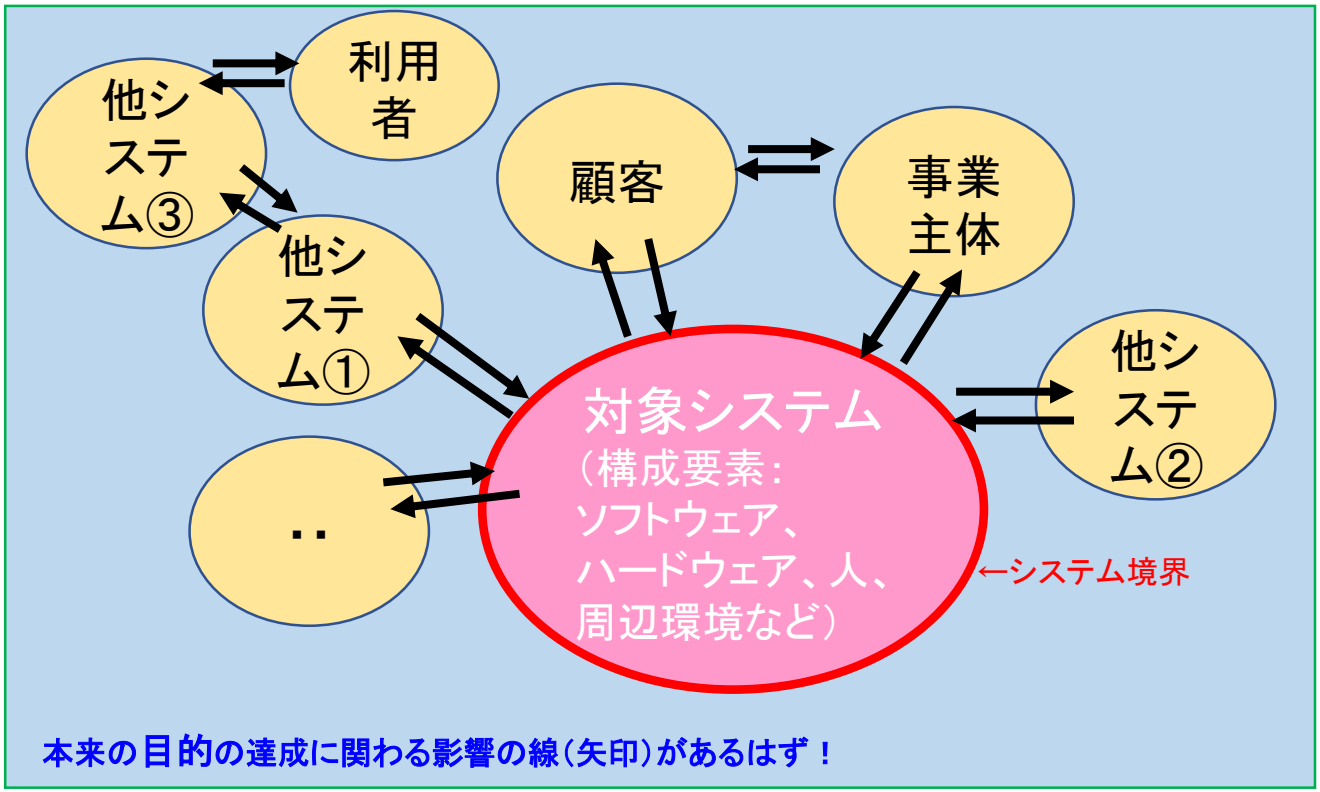


コンテキスト図の活用(例)

コンテキスト図で概観して妥当性を確認する

対象システムの振る舞いと、それを取り囲む**環境**との**相互作用**を表し

⇒本来の目的に対する妥当性を確認する



【相互作用の例】

- 人の行う操作
- 表示するメッセージ
- センサへのデータ取得指示
- センサからの取得データ送信

※ITシステムとしてのインターフェースのことではない

【コンテキストの範囲】
本来の目的を考える上で必要な考えるべきコンテキストの範囲を設定する(狭めすぎない)