IPA Better Life with IT

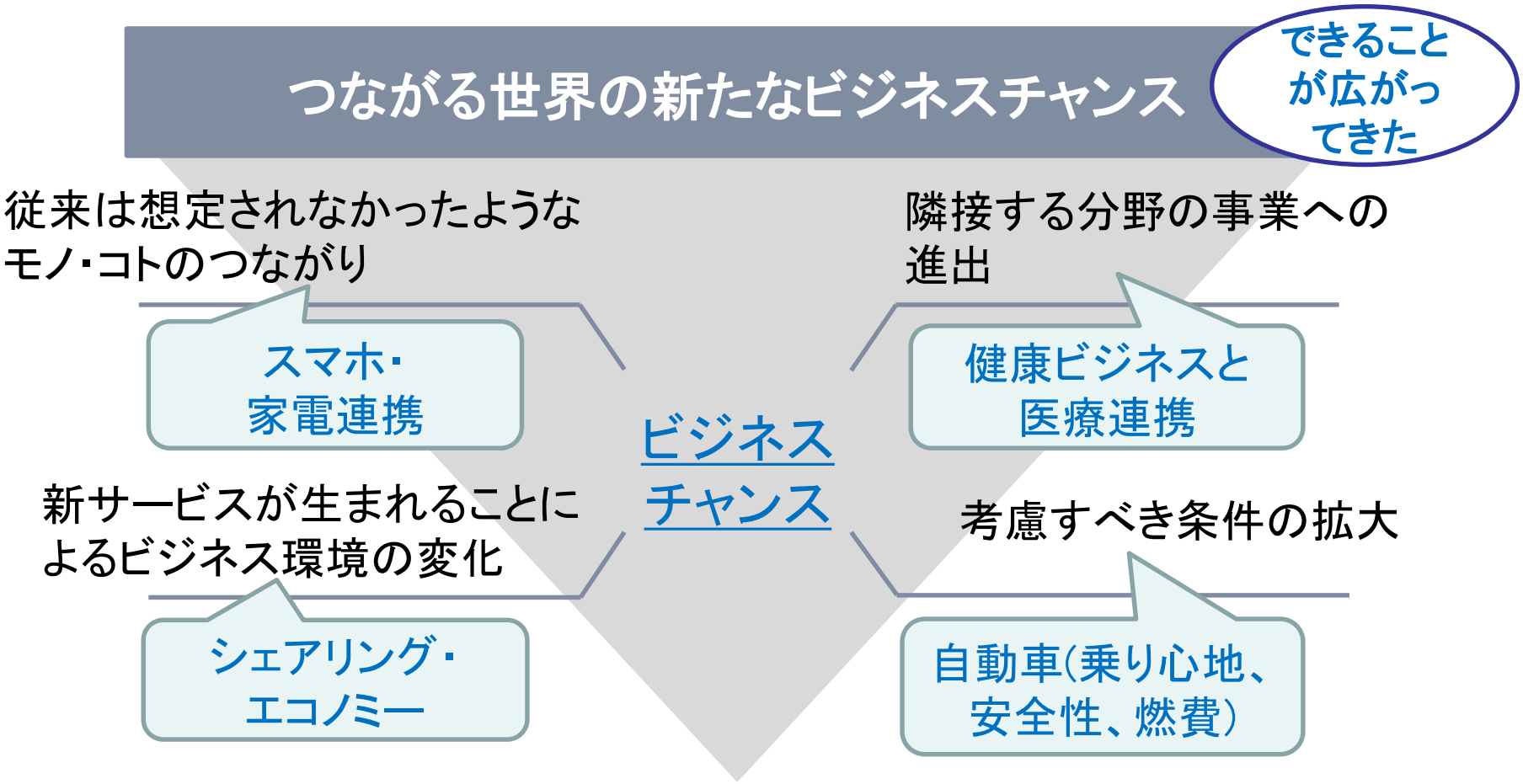
資料①

2019年3月19日　IPAセミナー
「システムズエンジニアリングを体感的に理解しよう！～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概説

独立行政法人情報処理推進機構(IPA)
社会基盤センター(IKC)

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

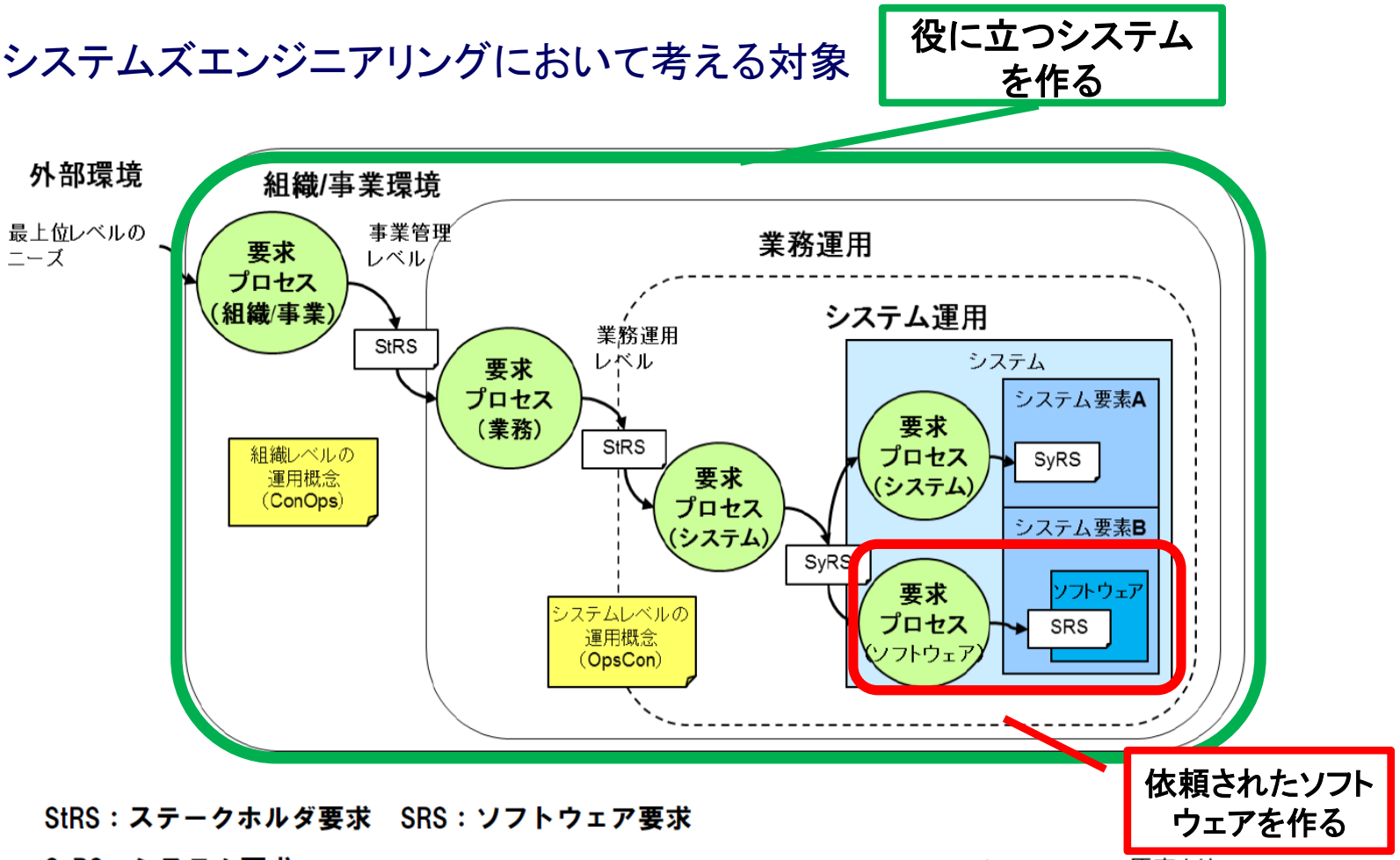
考慮すべき条件の拡大

- 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

考慮の対象

システムズエンジニアリングにおいて考える対象



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

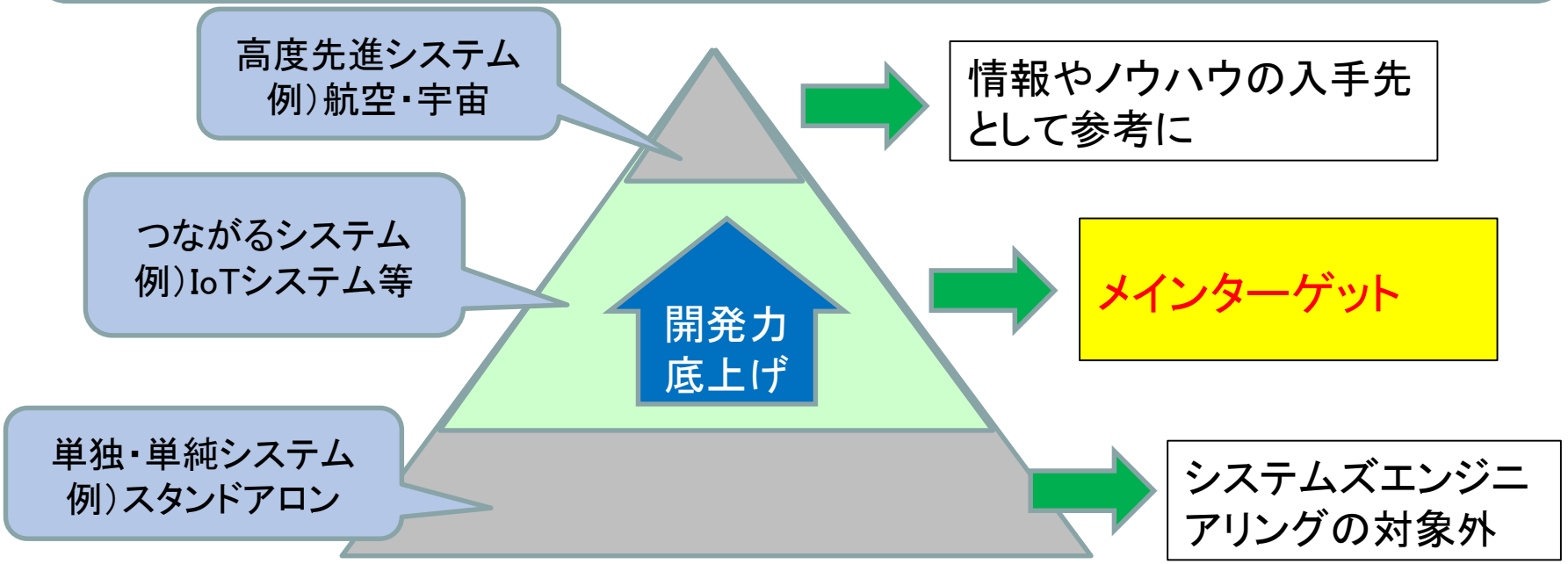
一段高い視点からの分析

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

→ これらはまさに「IoT時代のシステム開発アプローチ」の要件

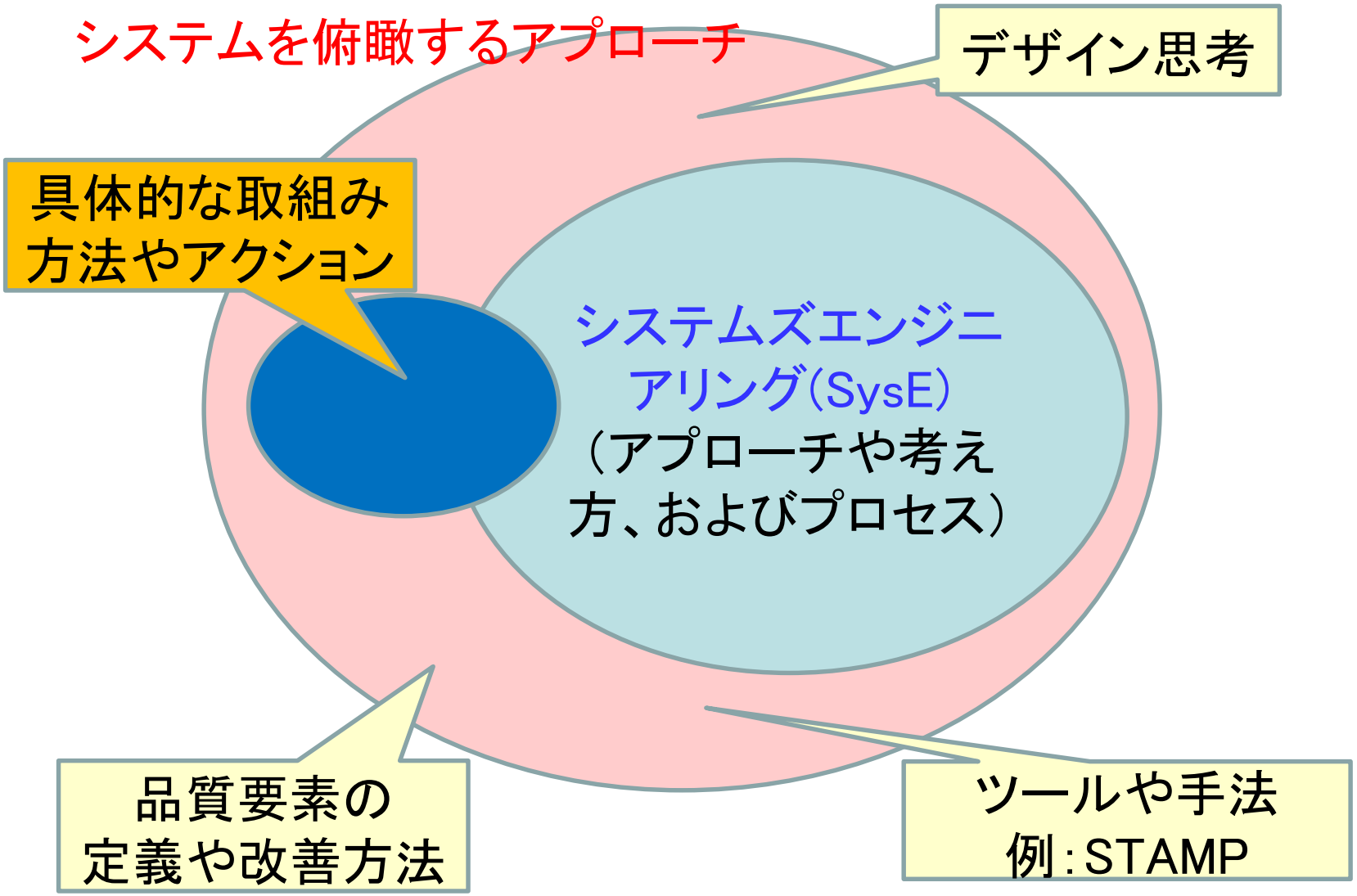
システムズエンジニアリングの展開ターゲット

- 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的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



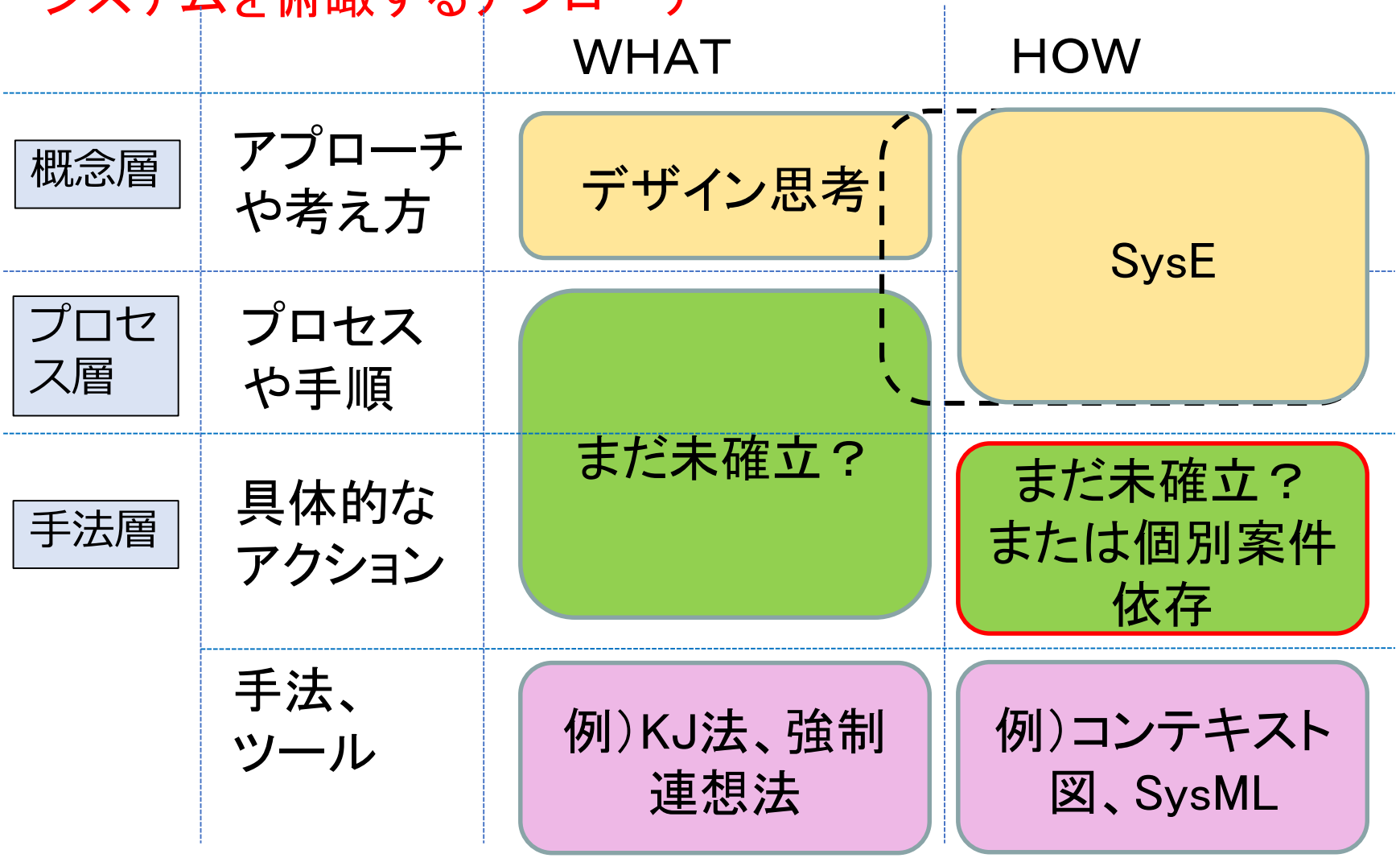
システムズエンジニアリングのメリット
- ◆ 協創：　多様の専門家、利害関係者による新たな製品・サービスの創造
- ◆ 考慮範囲：　複雑な「つながり」から生じる広範なリスク・問題への対応

システムを俯瞰するアプローチとSysE

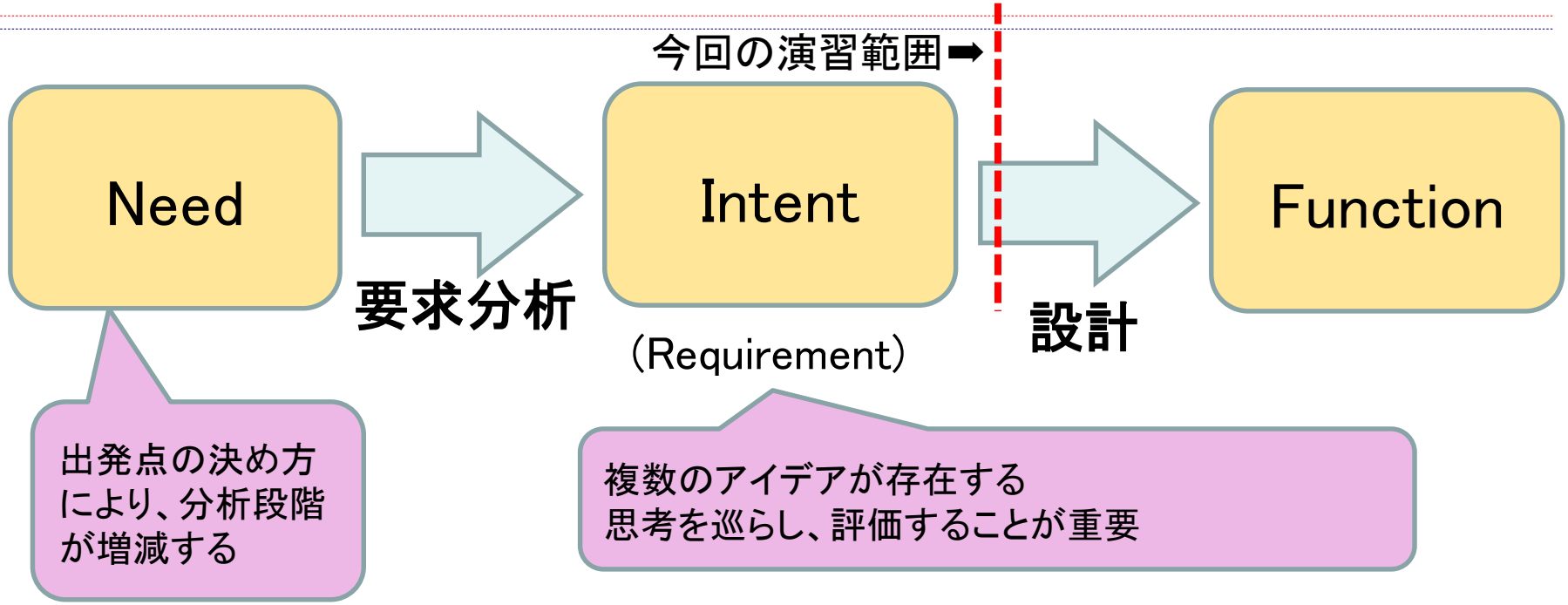


システムを俯瞰するアプローチとSysE

システムを俯瞰するアプローチ



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

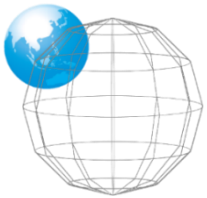
①目的指向と全体俯瞰

- 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。

出典：「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)

（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
⇒色々な立場から考えること
（自分本位な考え方を排斥し、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

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)

（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。

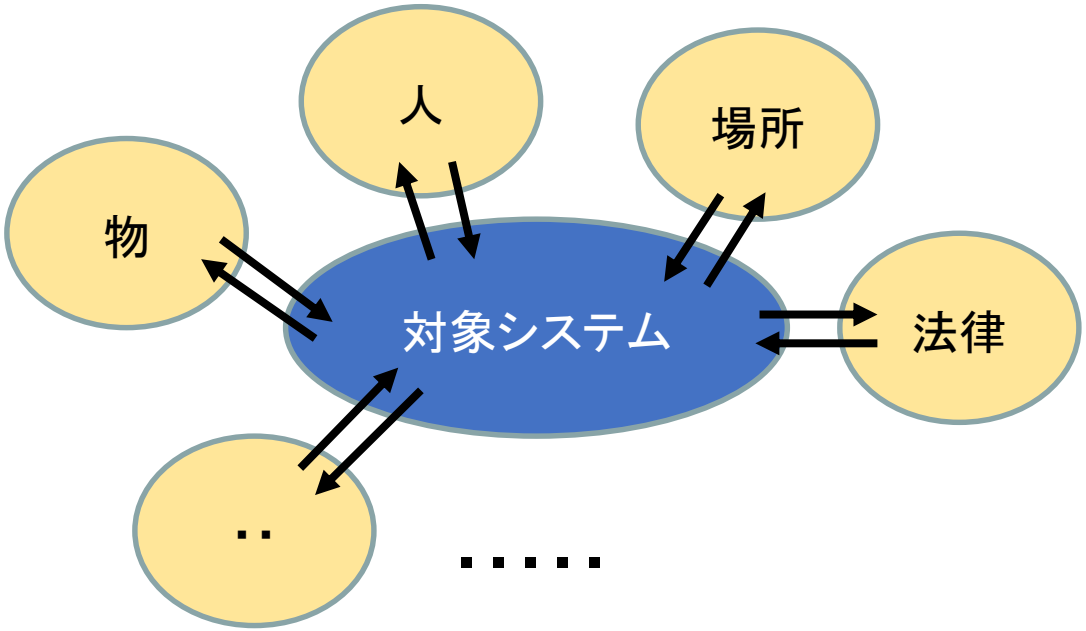
開発時、出荷後の初期設定時、利用時、休止時、更改時、破棄時

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

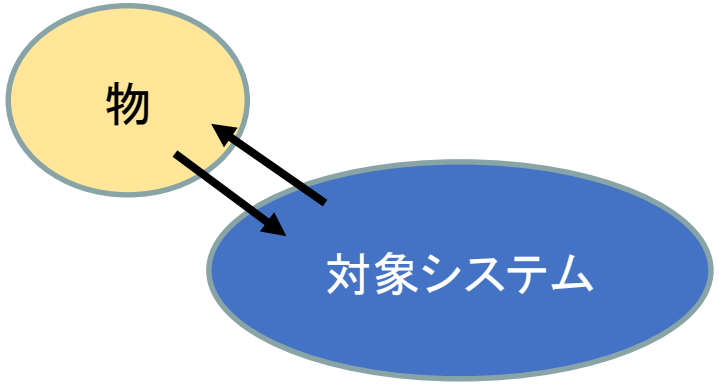
〇センサからの
取得データ送信

対象システムの実装に踏み込む前に、外堀を埋める

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。

このように考えを巡らすことが、気づきや考え漏れ防止に

先入観や思い込みに注意

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

〇統合・意識合わせは、作ってからでは調整負担が大きい

⇒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。

(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

○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

○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

○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）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

○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

○開発側は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。

テクニカルプロセス概観

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

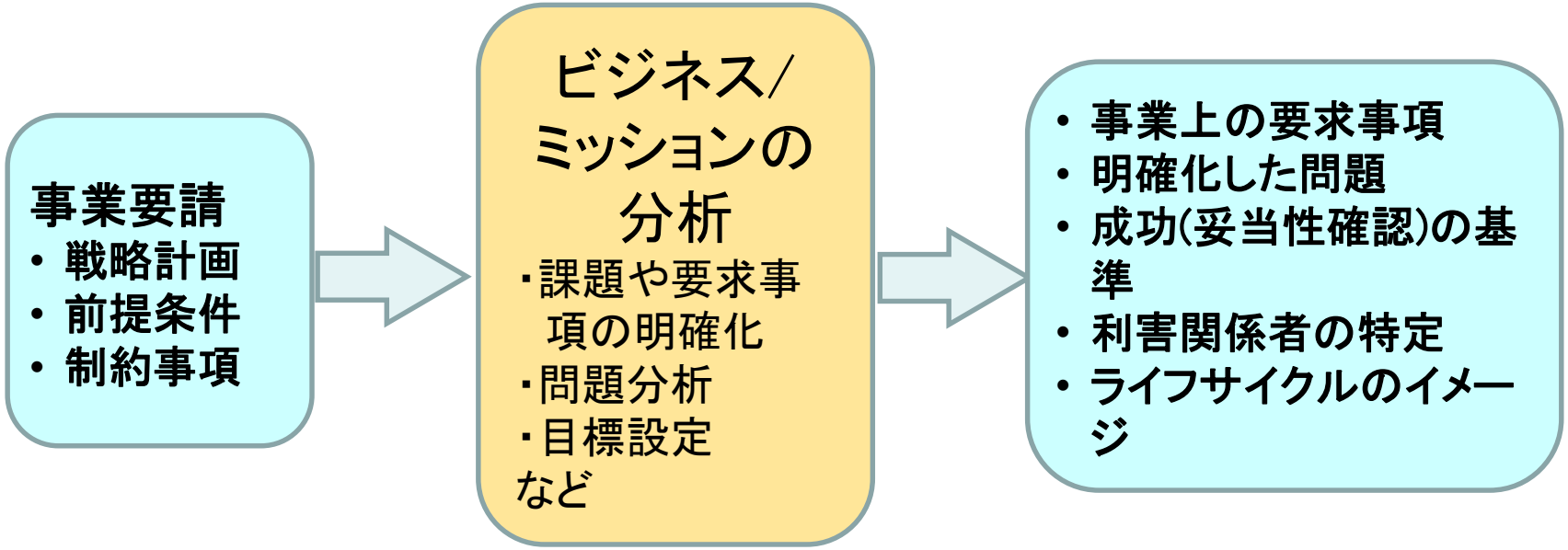
部分について、以下で詳説

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

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】

背景　野菜をたくさん食べれば、元気で賢い子になるはず

問題

自分の5歳の子供が野菜嫌いで食べない

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。

ゴール
自分の5歳の子供が野菜を食べる

6．4．1　ビジネスあるいはミッションの分析

ここでは、ビジネス/ミッション分析における留意点を示す。

出発点（今回の目的）の明確化

○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

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。

6．4．2　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

利害関係者の要求事項

- 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。

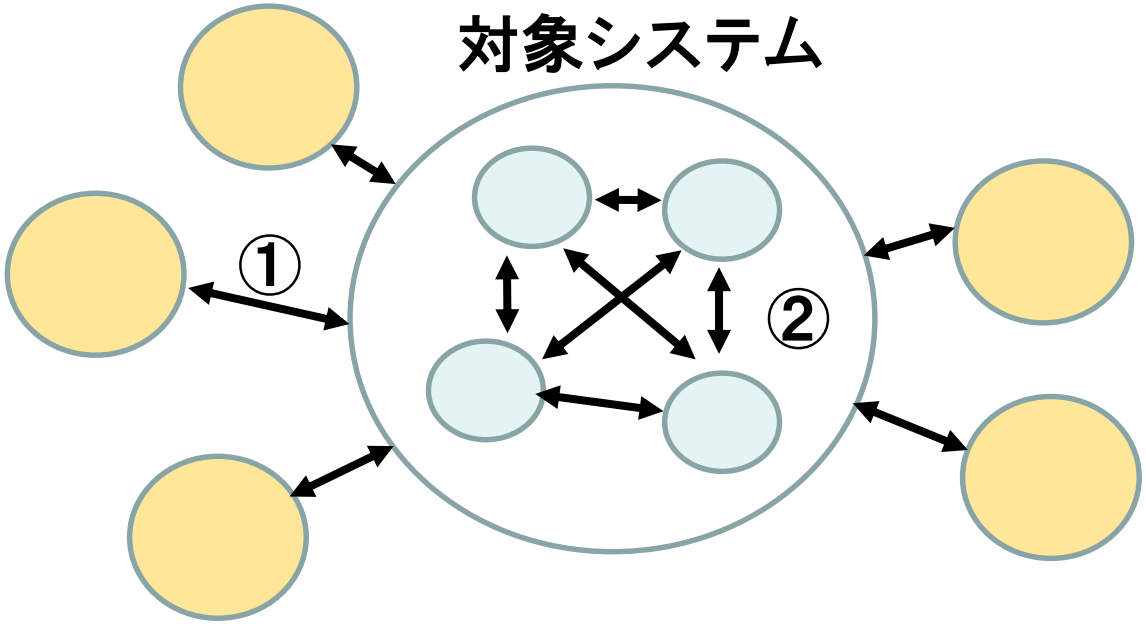
6．4．2　利害関係者ニーズと要求事項の定義

ここでは、利害関係者ニーズと要求事項の定義における留意点を示す。

Black Box Requirement　であること

○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



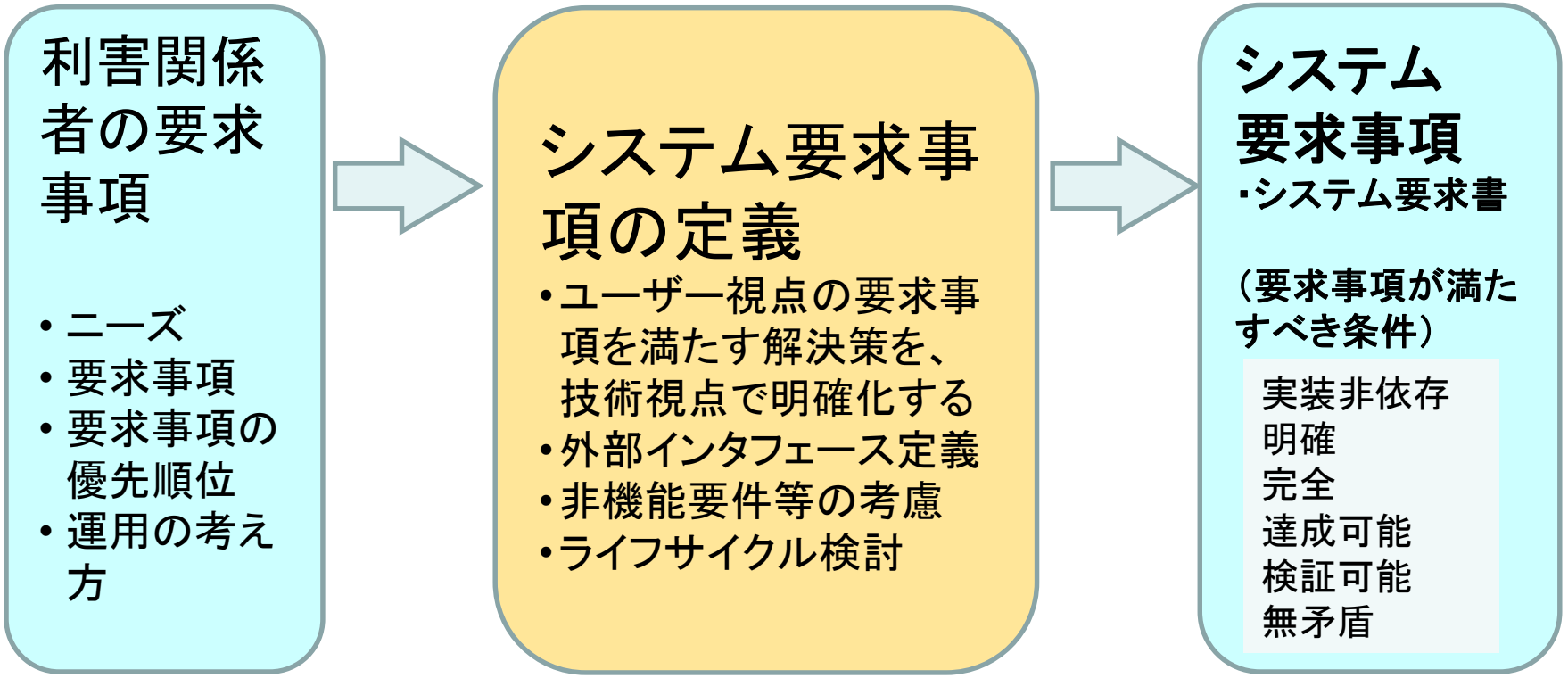
①対象システムと外界との関係
=Black Box Requirement
②システムを構成する要素と
構成要素間の関係

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。

6．4．3　システム要求事項の定義

ここでは、システム要求事項の定義における留意点を示す。

システム提供側のWILL（方針）を明確化

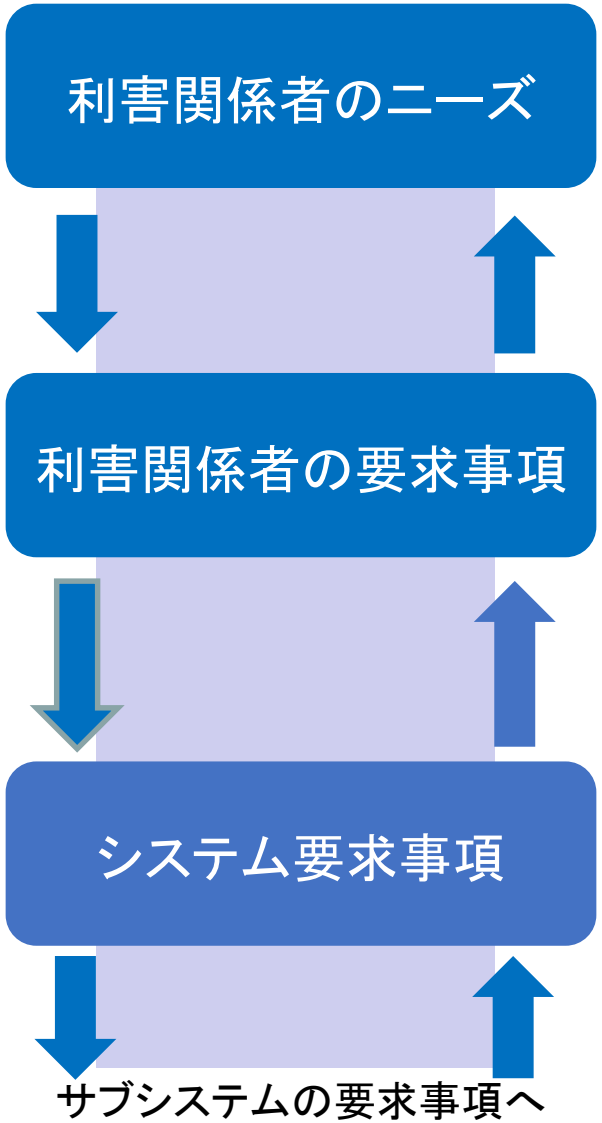
〇利害関係者ニーズ等の外部要件を全て実現することは不可能
⇒ニーズに応える事項の優先度付けを、システム提供側のポリシーとして示し、どこかに一線を引いて、実現する範囲を明確化

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



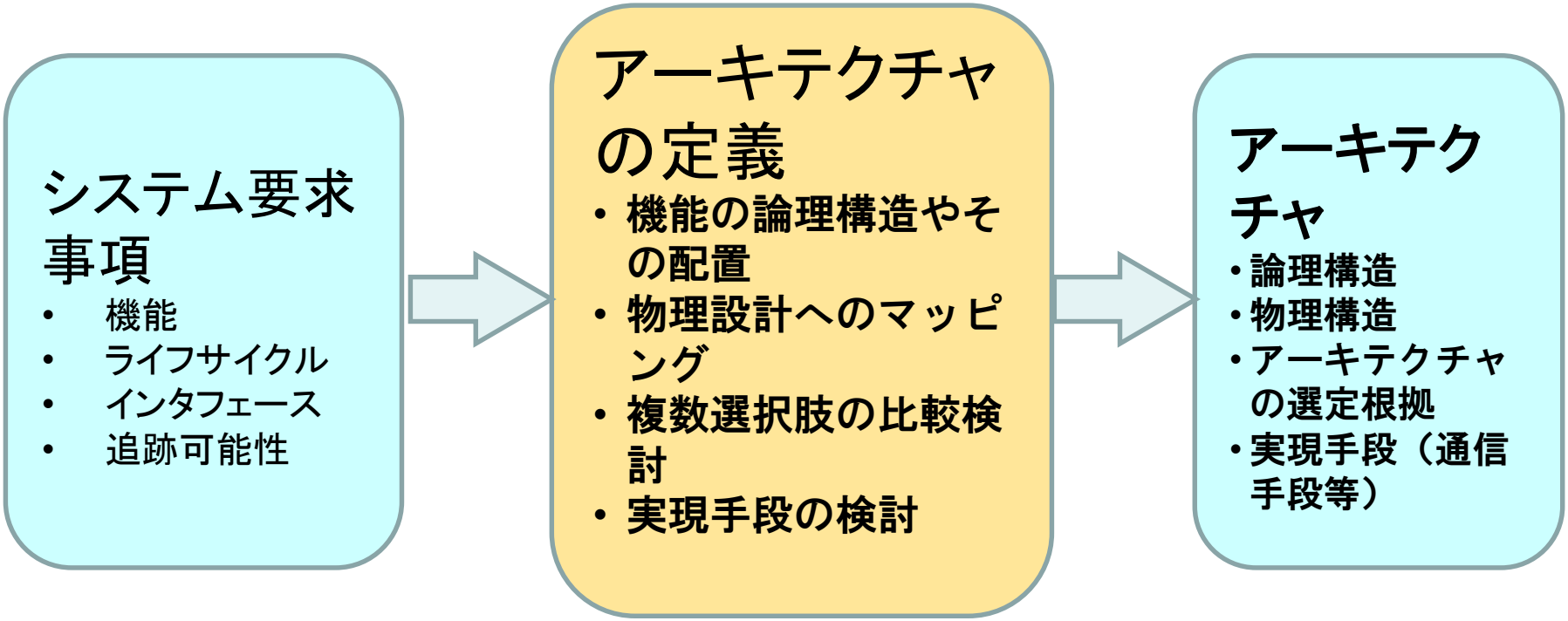
- 利害関係者の言葉で、問題意識や期待が表現される
- 曖昧、不正確、達成判断不能な場合もある

- 利害関係者のニーズを取捨選択して、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

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、独自の解釈が含まれています。

6．4．4　アーキテクチャの定義

ここでは、アーキテクチャ設計の留意点を示す。

機能と物理の分離

○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

○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

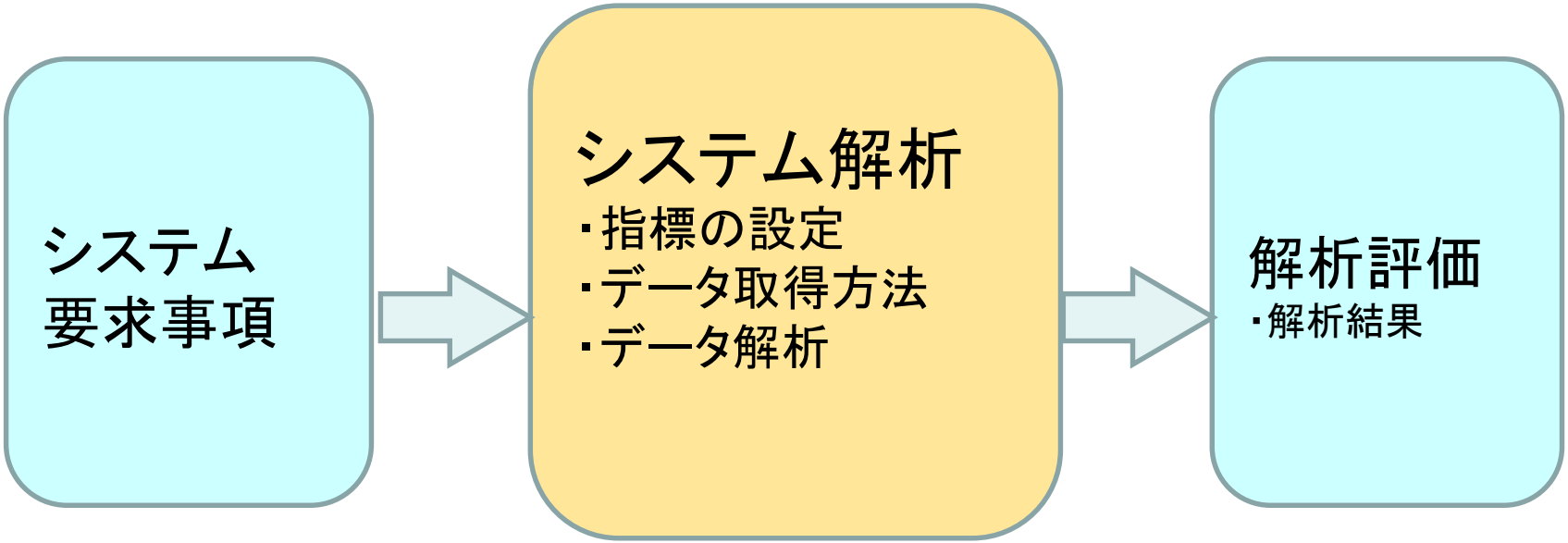
実現手段の比較検討

○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、独自の解釈が含まれています。

6．4．6　システム解析

ここでは、システム解析において留意すべき点を示す。

システムの技術的特性を指標定義、分析

○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

○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（○○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）

○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】

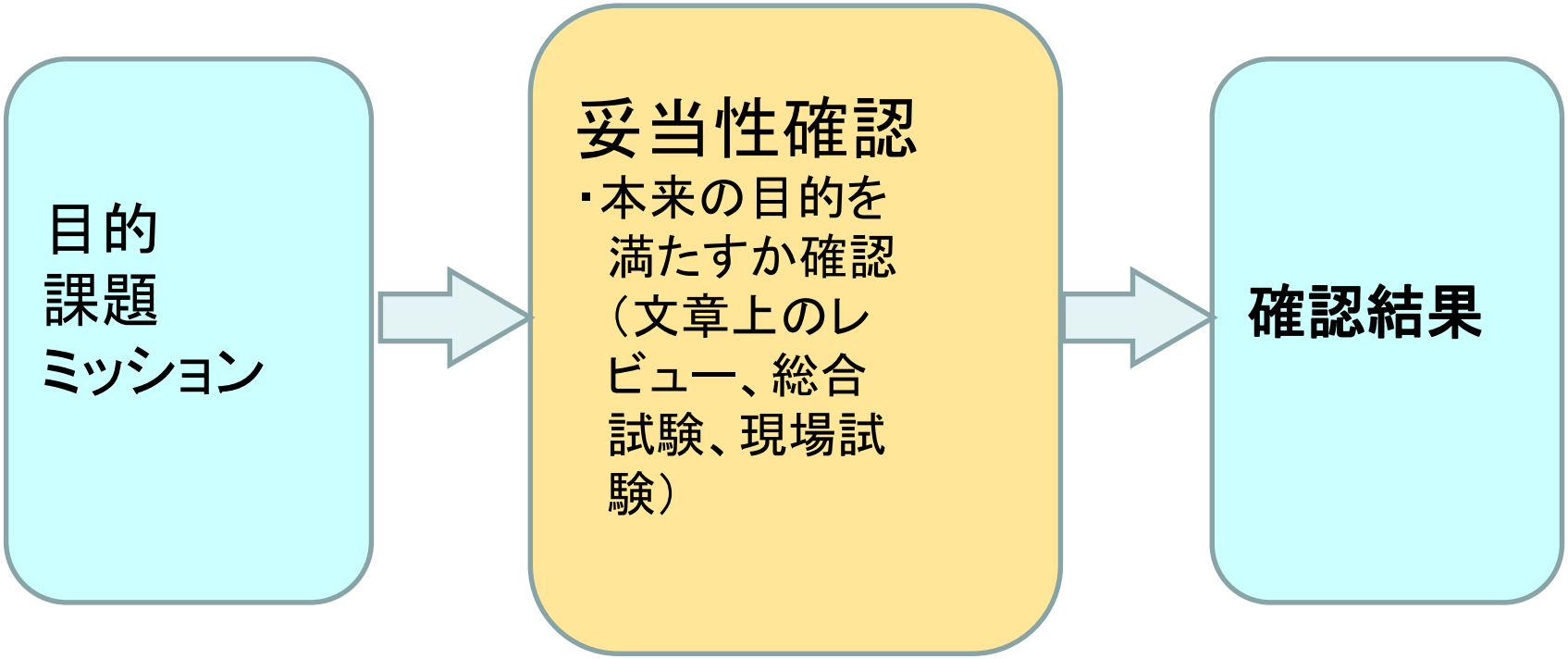
・顧客データが破壊された時、システム以外に復旧手段がない。
・顧客データが盗まれると、会社責任が問われ、存続が危ぶまれる。
・システムが止まっても、今まで手仕事でやっていたことをシステムで自動化しただけなので、最悪でも運用でカバー出来る。

【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。

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。

6．4．11　妥当性確認

ここでは、妥当性確認における留意点を示す。

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。

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」との交点で対策を練る

プロセス ＼ ポイント		ポイント				
		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】					
	(6.4.2)【利害関係者ニーズと要求事項の定義】					
	(6.4.3)【システム要求事項の定義】					
	(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】					
	その他					