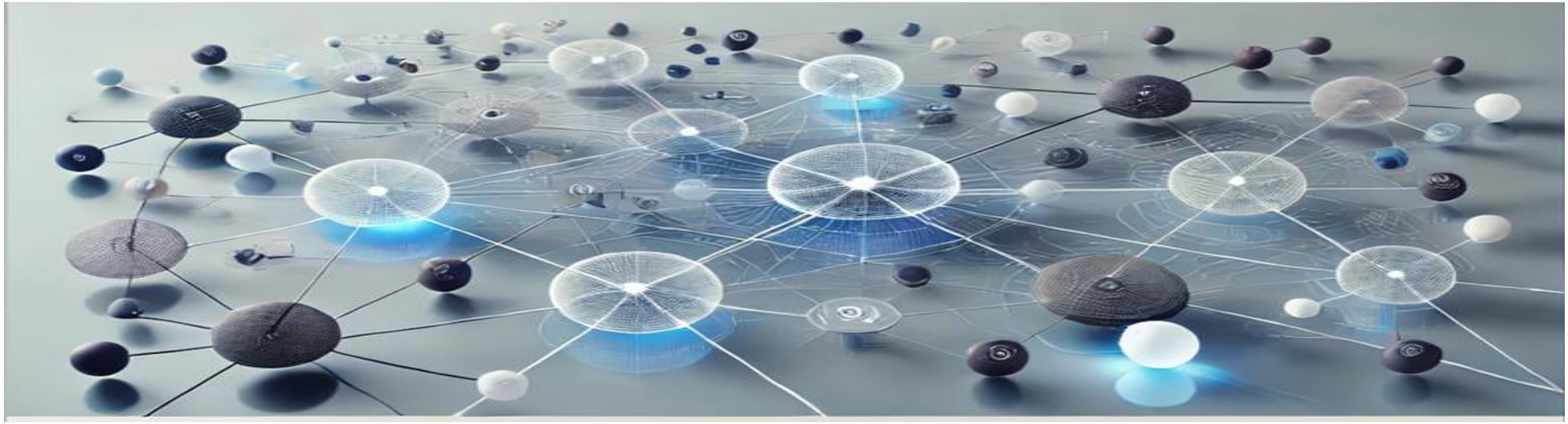
第67回P2Mクラブ（2025年3月27日）

KSP-25-001
2025/3/27

ビジネスを駆動するプロジェクトマネージャにとってのシステムズエンジニアリング

～ディシプリン：その規範的な力～

檜垣造船株式会社　佐藤　健司

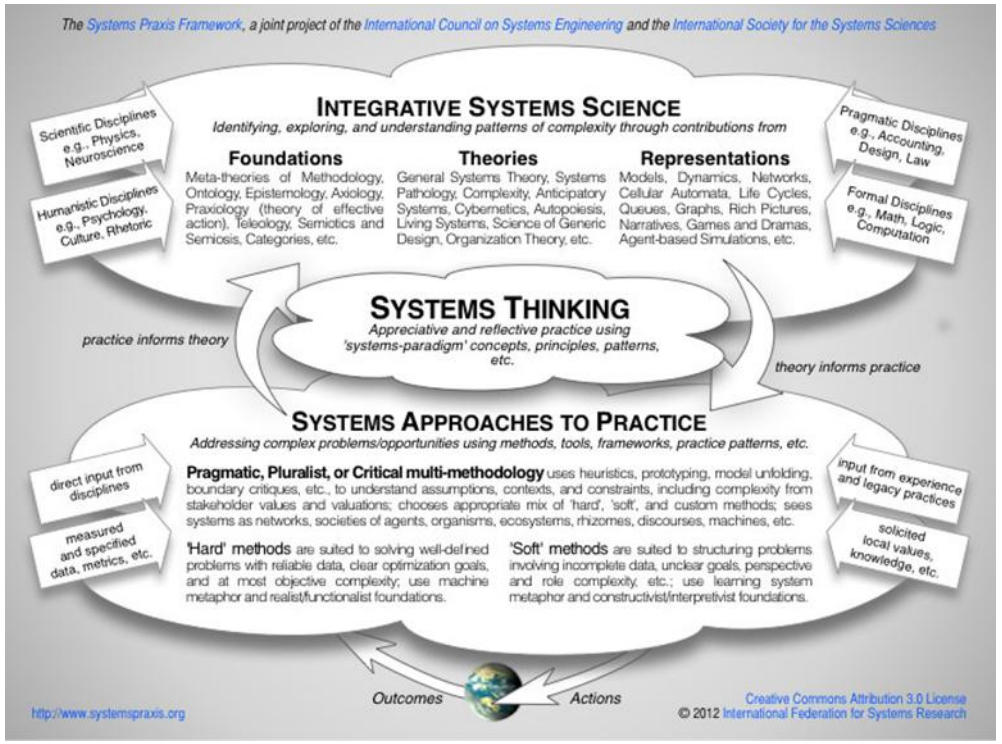


第二部

システムズエンジニアリング.... *SO WHY? SO WHAT?*

1. その意味
2. 駆動する力
3. 規範的な力
4. ディシプリン
5. 適用の滞り
6. コンテクストの理解
7. その意思決定

本日のテーマ



出典：SEBOK

はじめに

永続する偉大な組織をつくるためには、

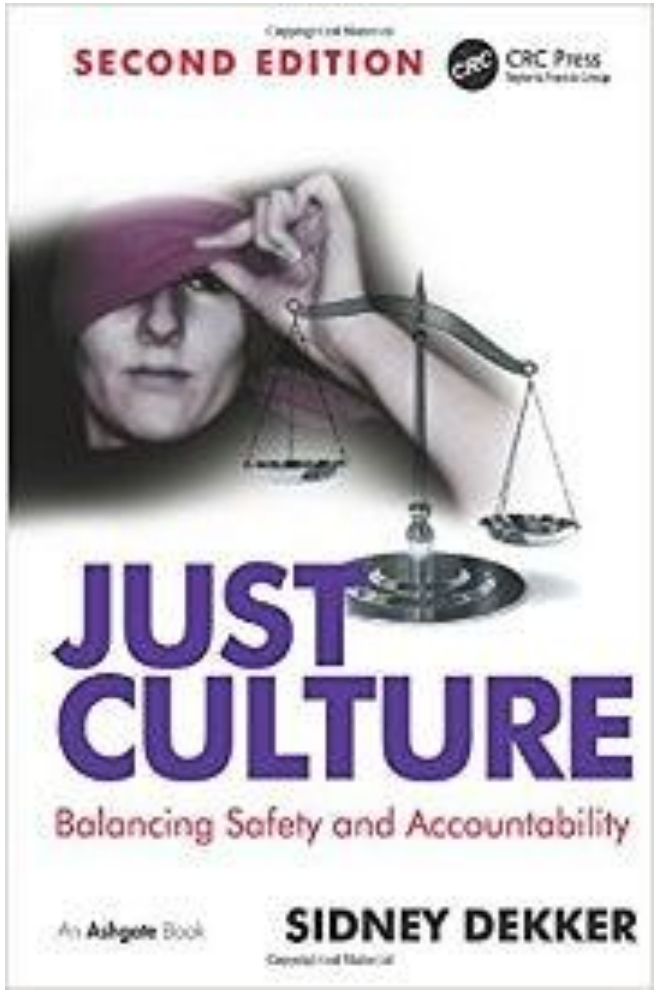
規律ある思考をし、

規律ある行動をとる

規律ある人材が必要だ。

（出典：日経BP発行　ジム・コリンズ、ビル・ラジアー著　ビジョナリーカンパニーZERO

システムズエンジニアリングの規範的な力



規範的な力の働き:

システムズエンジニアリングでは、規範や基準が設定され、それに従って行動することが重要です。これによって、プロジェクトや組織が効果的に機能し、目標に向かって進むことができます。しかし、順調に進行している場合は、この規範的な力が目立たないことがあります。

混乱やミスの時の対応:

システムズエンジニアリングにおいては、混乱やミスが発生したときには、規範や基準に従うことが重要です。これによって、問題が解決され、プロジェクトの進行がスムーズになります。

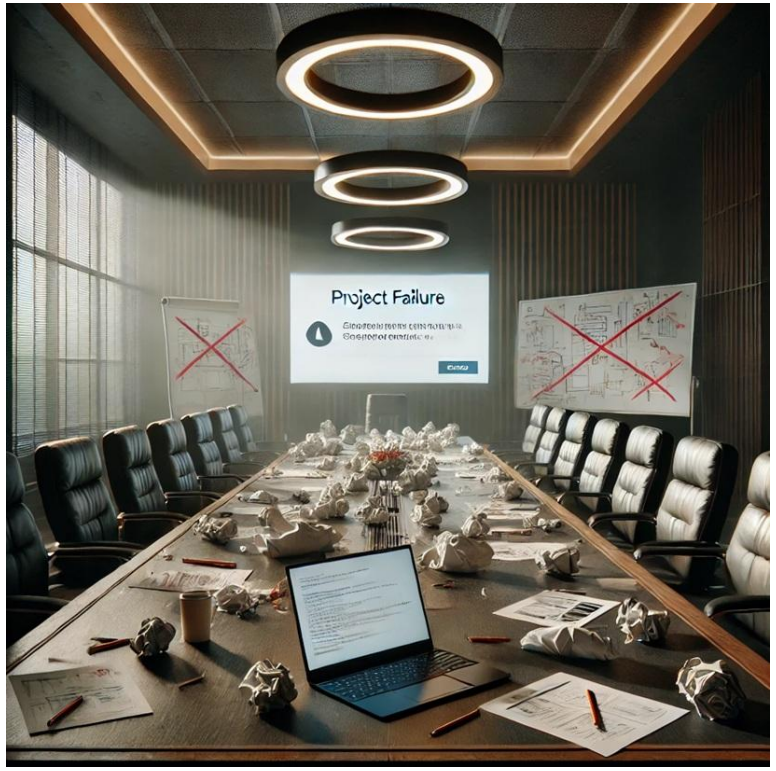
規範的な力は、こうした混乱やミスの時にピンポイントで働くことがあります。

自然な従順:

一方で、順調に進行している場合は、規範的な力が目立たないかもしれません。 このような場合、プロジェクトや組織のメンバーは、自然に規範や基準に従って行動することがあります。

規範的な力が必要とされるのは、問題や混乱が発生したときであり、その時に働くことで効果的な対応が可能になります。

わたしたちが痛みを経験する理由



わたしたちが痛みを経験するには、理由がある。痛みは、危険を警告したり、対処するように知らせてくれる。そのプロセスが生存にとってとてつもない利点を与えてくれる（中略）。

内なる声の厳しい側面の必要性を映し出している。内なる声は、ネガティブな感情によって思考を曇らせることがあるものの、そのような批判的な自己省察がなければ、学び、変わり、向上することは難しいだろう。

（出典：東洋経済新報社発行　イーサン・クロス著　Chatter「頭の中のひとりごと」をコントロールし、最良の行動を導くための２６の方法

システムズエンジニアリングのディシプリン

「ディシプリン（規律や方法論）」という語は、システムズエンジニアリングにおいて基本的な原則やルールを教えるときや、違反をただすときに使用されます。

問題なくシステムズエンジニアリングの適用が行われている場合でも、実際にはディシプリンが遵守されていない場合もあります。

ディシプリンはプロジェクトの成功に不可欠であり、適切な適用が重要です。

1 SYSTEMS ENGINEERING HANDBOOK SCOPE

1.1 PURPOSE

This handbook defines the discipline and practice of systems engineering (SE) for students and practicing professionals alike and provides an authoritative reference to understand the SE discipline in terms of content and practice.

（出典；INCOSE発行　SYSTEMS ENGINEERING HANDBOOK FOURCE EDITION）

規律をもたらす仕組み

規律を志向する仕組みには2つのものがあると考えられる。それは、学習と推論だ。
私たちは、この学習と推論を組み合わせることによって、規律を確立し、それを維持している。日常のルーチンの多くは、最初から無意識に行えるわけではなく、何らかの試行錯誤（学習）を経て形成されたものであり、また、そのルーチンを選択する際には、何が正しいかを考え（推論）、確立した可能性がある。一度、ある程度満足できる規律に落ち着けば、それは無意識に近い形で継続されるが、環境の変化や新たな状況が生じた際には、再び試行錯誤や再評価が必要となる。つまり、「無意識にできる」という状態は、規律の営みが安定している間の一時的な状態に過ぎず、状況に応じた適応と改善を続けることが、本質的な規律の在り方である。

学習とは、何かを実行した後にその結果を評価し、次に同じことをする際に改善を加えながら最適化していく仕組みである。これは、規律を持続させる上で重要な要素であり、試行錯誤を通じて、無意識に近い形で規律ある行動が定着するプロセスである。

推論は、何かを行う前に、一定の基準に基づいて評価し、最も適切な選択肢を見極める仕組みである。組織や個人が規律を保つためには、単なる反復ではなく、その行動が適切であるかを常に評価し、必要に応じて修正を加える思考が求められる。

「（参考文献：青弓社発行　佐藤　裕著　ルールの科学」のコンテクストを規律へと転用

システム複雑性とシステムエンジニアリング（SE）の課題

システムの特性（客観的複雑性）

- 多くの要素（Many pieces）
- 非線形（Nonlinear）
- カオス的（Chaotic）
- 適応性（Adaptive）
- 密接に結合されている（Tightly coupled）
- 自己組織化（Self-Organized）
- 分散型（Decentralized）
- 政治的（Political）
- 開かれた（Open）
- 多スケール（Multi-Scale）
- 創発（Emergent）

複雑性の客観的側面と主観的側面を区別し、それぞれに適切な対応が求められる

認知的特性（主観的複雑性）

- 不確実（Uncertain）
- 理解が難しい（Difficult to understand）
- 因果関係が不明瞭（Unclear cause and effect）
- 予測不可能（Unpredictable）
- 不安定（Unstable）
- 制御不能（Uncontrollable）
- 修復・維持が困難（Unrepairable, unmaintainable）
- 建設に時間がかかる（Takes too long to build）
- 高コスト（Costly）

出典；INCOSE　complexity-primer-overview

規範が必要となる理由

規範がなければ、混乱や不正が生じやすくなります。そのため、以下のような問題となる状況が生まれます。

好ましくない状態	意味	発生する問題
Anarchy（無統治）	権威や支配が存在せず、統治が機能していない状態	明確なルールや指導がなく、個々の判断に依存するため、組織や社会がまとまりを失う
Chaos（混沌）	構造や秩序がなく、不確実性や混乱が支配する状態	予測不能な出来事が頻発し、計画や戦略が立てられず、意思決定が困難になる
Lawlessness（無法）	法律や規則が守られず、制約がない状態	犯罪や不正が増え、社会的な信用が損なわれる
Arbitrariness（恣意性）	一貫した基準がなく、個人の判断や気まぐれで決定される状態	公平性や客観性が失われ、不平等や不信感を生む
Deviance（逸脱）	社会や組織の一般的なルールから逸脱する行動や価値観	社会的摩擦が生じ、組織の一体性が損なわれる

文脈（コンテクスト）によって異なる意味(1/2)
～ディシプリン：その規範的な力～

	文脈による異なる意味	内容
1	規律や統制 （規律的な側面）	自己管理や集団での規則を守る行動を指します。この文脈では、秩序や効率を維持するための行動や態度を含みます。
2	学問や専門分野 （知的な側面）	NASA Systems Engineering Handbookでは、特定の学問や研究分野を指します。この場合、知識体系や研究対象、方法論を共有するコミュニティが関わります。
3	鍛錬や修練 （能力育成の側面）	ピータ・M・センゲ著Fifth Discipline（システム思考）にあるように、スキルや能力を向上させるための訓練や努力を指します。この文脈では、一定の目的に向けて継続的に取り組むことが重要です。

規律や統制（規律的な側面）：標準

IEEE Standard	Description (IEEE)	JIS Standard	Description (JIS)
IEEE 1220	Systems Engineering Process	JIS X 0170:2020	システムライフサイクルプロセス
IEEE 1471 (ISO/IEC/IEEE 42010)	Systems and Software Architecture Description	JIS X 0166:2014	ソフトウェアライフサイクルプロセス
IEEE 15288 (ISO/IEC/IEEE 15288)	System Lifecycle Processes	JIS X 0170:2020	システムライフサイクルプロセス
IEEE 29148	Requirements Engineering Standard	JIS X 0166:2014	システム及びソフトウェア技術－ライフサイクルプロセス－要求工学
IEEE 2675	DevOps in Safety-Critical Systems	N/A (No direct JIS equivalent)	

学問や専門分野（知的な側面）
複数の分野を横断的に調整しながら設計を進める

システムズエンジニアリングでは、特定の専門分野が単独で支配的にならず、複数の分野を横断的に調整しながら設計を進めること が求められます。このため、**「discipline」という言葉が、単に「学問」や「技術領域」以上に、「専門家の知見が交差し調和する場」という意味を持っています。**

Holistic（全体的な）	システムズエンジニアリングは、個別の要素だけでなく、システム全体を見渡して設計する。
Integrative（統合的な）	各分野の専門知識を統合し、最適な解決策を見つける。
Discipline（専門分野）	構造工学、電気工学、人間工学など、異なる技術領域。
Balanced, one against another（互いにバランスをとる）	どの分野にも偏らず、全体最適を目指す。

HOLISTIC（全体的な）

ある特定の専門分野だけの視点ではなく、システム全体を見て最適化する

システムズエンジニアリングは トレードオフ（Trade-offs）と妥協（Compromises） を扱うプロセスであり、「システム全体を横断的に見る視点（broad crosscutting view）」を重視する

Broad crosscutting view of the system

（システム全体を横断的に見る視点）

- システム全体のパフォーマンス、機能、コスト、スケジュール、安全性などを考慮し、最適なバランスをとる。
- 個々の専門分野に閉じず、異なる分野間のトレードオフを行う。

Single discipline view

（単一の専門分野の視点）

- ある特定の専門分野（例：電気工学、機械工学、人間工学）の視点だけで最適化しようとする。
- その結果、システム全体のバランスを損なう可能性がある。

INTEGRATIVE（統合的な）

特定の技術領域にとらわれずに横断的に影響を与える

効果的なプロセスの実現には、異なるスキルを持つ専門家が協力し、オープンな姿勢を持つことが重要である。特にCrosscutting Specialty Discipline（横断的な専門技術分野）は、システム全体のバランスを考慮しながら、特定の技術領域にとらわれずに横断的に影響を与える。

- 信頼性工学（Reliability Engineering） → システムの信頼性向上
- 安全性工学（Safety Engineering） → システムのリスク管理
- 保守性工学（Maintainability Engineering） → メンテナンスのしやすさを考慮
- 統合ロジスティクス支援（Integrated Logistics Support, ILS） → 運用支援とサプライチェーン管理
- 人間工学（Human Factors Engineering） → ユーザーの使いやすさの最適化

DISCIPLINE（専門分野）

専門技術分野も関与して作業を行う

Specialty engineering disciplines（専門技術分野） とは、システムズエンジニアリングの中でも特定の目的に特化した技術分野を指します。

Maintainability（保守性工学）

- システムや製品がどれだけ容易に修理・保守できるかを設計する分野。
- 設計段階で、故障時の修理時間を短縮したり、メンテナンスの負担を軽減することを考慮する。

Logistics servicing（ロジスティクス支援）

- システム運用中の補給・部品交換・修理サポートなどを最適化する分野。
- 例えば、航空機や軍用システムでは、修理部品の供給や保守計画を事前に設計する必要がある。

BALANCED, ONE AGAINST ANOTHER（互いにバランスをとる）

異なる専門技術分野や設計チームを統合し、バランスをとる

異なる技術分野の専門家（discipline teams）が、それぞれの視点から設計を行うため、システム全体のバランスが崩れるリスクがある。そこで、システムズエンジニアが中心となり、それらを統合し、適切なバランスを取る役割を果たす必要がある。システムズエンジニアの役割は、異なる専門技術分野や設計チームを統合し、システムと環境の相互作用が適切にバランスされるようにすることである

	ポイント	内容
1	Integration（統合）	システムズエンジニアの役割は、異なる専門技術分野（discipline）と設計チーム（design teams）を調整し、統合すること にある。
2	Balancing system and environmental interactions（システムと環境の相互作用のバランスを取る）	システムが周囲の環境（物理環境、運用環境、安全要件など）とうまく調和するように、異なる分野の設計チームが適切に連携する必要がある。
3	Differing design teams（異なる設計チーム）	各専門技術分野の設計チームは、それぞれの技術に特化しており、必ずしもシステム全体を考慮しているわけではない。システムズエンジニアは、それらの設計チームを調整し、最適なトレードオフを実現する。

確立された管理手法としての適用

Management discipline（管理手法） とは、単なるルールではなく、組織的に確立された管理プロセスや実践体系 を指す。例えば、構成管理（CM: Configuration Management）は、以下のような目的で適用される。

	ポイント	内容
1	Visibility into changes（変更の可視化）	製品の性能、機能、物理的特性の変更が、どのように管理されているかを明確にする。
2	Control of changes（変更の制御）	変更が無秩序に行われるのを防ぎ、適切なプロセスを経て管理する。
3	Applied over a product’s life cycle（製品ライフサイクル全体に適用）	設計、開発、製造、運用、保守のすべてのフェーズで適用される。

構成管理（CM: Configuration Management）については、JAXA発行「JMR-006（コンフィギュレーション管理標準）」が活用可能です。

システムズアプローチ　規律ある、体系化された、計画的な方法論

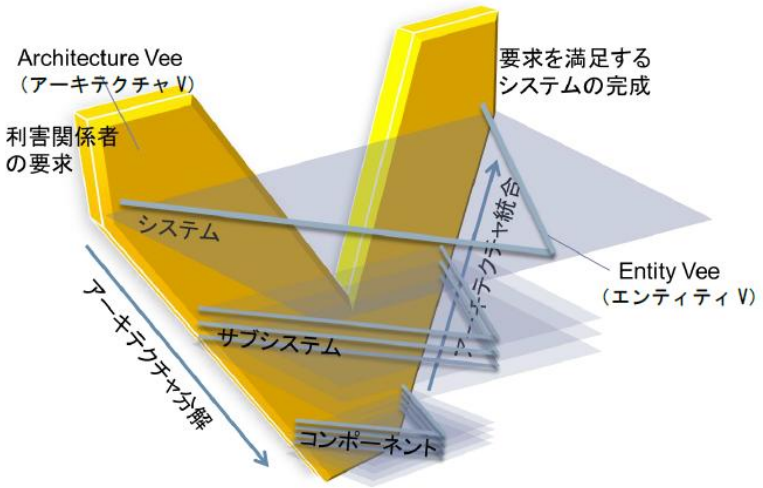
システムズアプローチは、計画的で定量的、かつ反復可能な方法論に基づいて、システムの開発・運用・維持に適用されるべきである。

すべてのシステムは、ニーズの認識、または、機会の発見から始まり、プロジェクトの終了に向けてさまざまな開発段階を経て進行する。システムズエンジニアリングに関連する分析や最適化活動の最も大きな影響は、初期段階で得られることが多いが、コストに影響を与える意思決定は、システムの寿命が終わりに近づいても、依然としてシステムアプローチによって最適化できる。

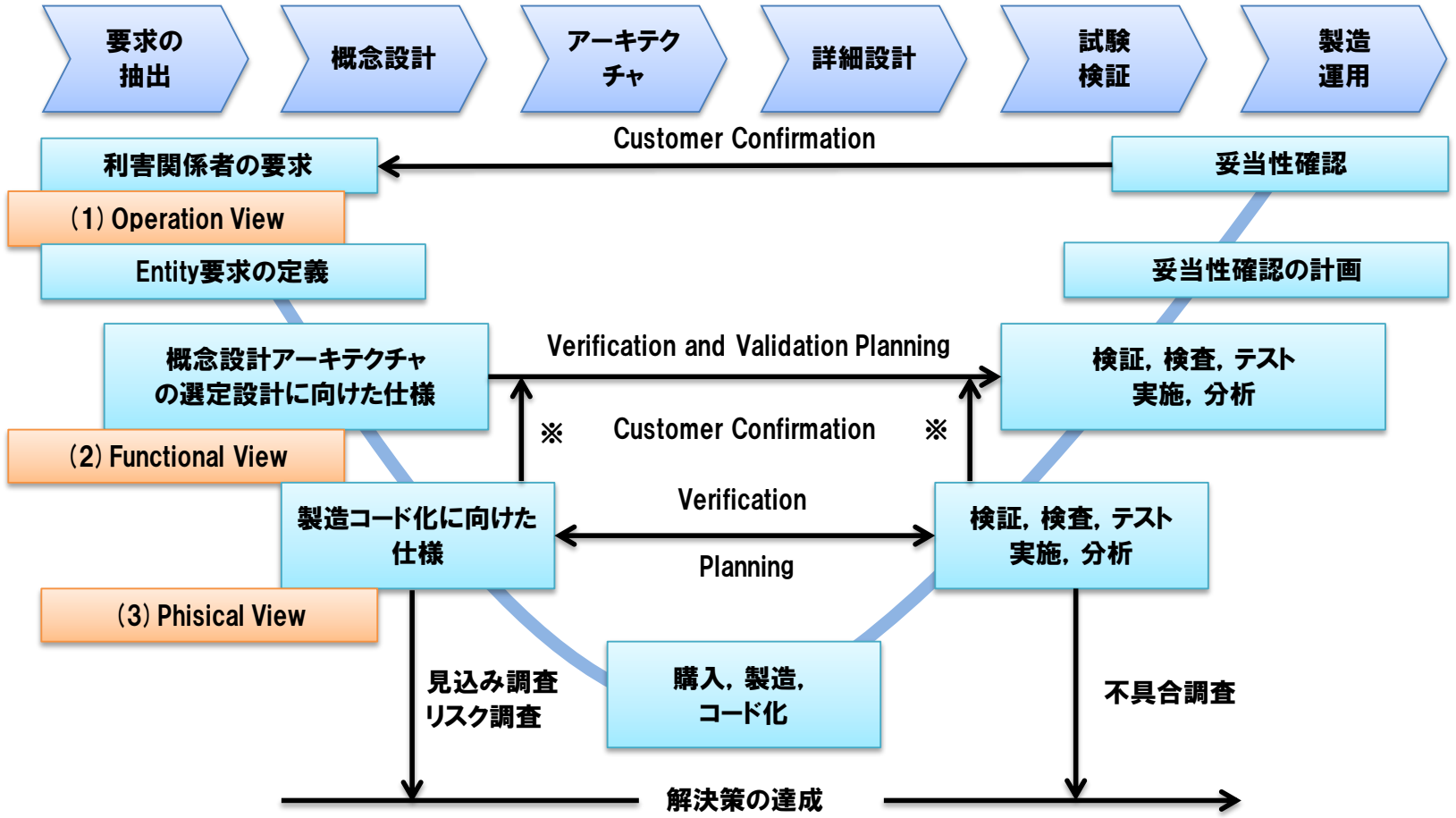
出典：NASA Systems Engineering Handbook

	ポイント	解説
1	disciplined は「規律ある」「体系的な」アプローチを意味する	単なる経験則や場当たり的な対応ではなく、明確なプロセスに基づいて実施されるべきことを示している。
2	システムズアプローチは、定量的・再帰的・反復的・再現可能であることが重要	これはシステム開発や運用の品質を高め、効率的に進めるための原則。
3	システムのライフサイクル全体に適用され、統合的に管理され	システムを全体最適の視点で捉え、開発・運用・保守において継続的に適用される。

反復による発見と進化



出展：IPA/SEC
モデルベースシステムズエンジニアリング導入の手引き



トップダウン／ボトムアップ：システムアーキテクチャとその実装・統合の方向性

システムの目的からスタートし、利害関係者の要求をもとに、コンセプトやハイレベルのアーキテクチャを設計するアプローチをトップダウン（Top-Down）と呼ぶ。この方法では、システム全体を「システム → サブシステム → コンポーネント」へと階層的に分解し、必要な要件や設計を具体化していく。

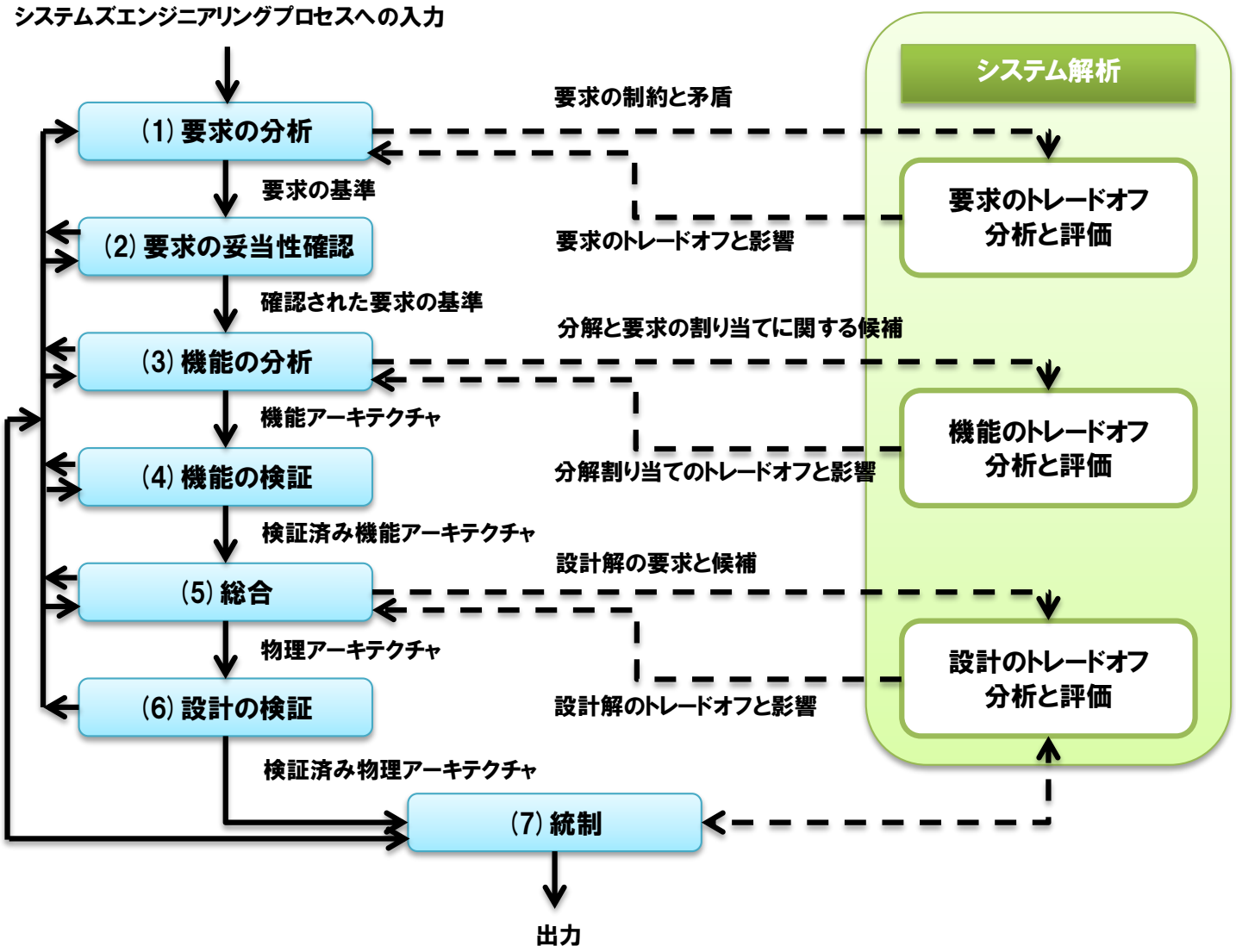
このプロセスはアーキテクチャV（Architecture Vee）の左側の流れに沿っており、システムズエンジニアリングのVモデルの左手側と一致する。

一方で、これに対する手法がボトムアップ（Bottom-Up）である。ここでは、最も低い階層のコンポーネントレベルからスタートし、利用可能な人工物、能力、またはサービスを組み合わせながら、創発的な特性を考慮しつつ、より高いシステム階層へと統合していく。

このアプローチはエンティティV（Entity Vee）の右側の流れに沿っており、各コンポーネントやサブシステムが統合され、システムの完成へと至る。

この2つのアプローチは相補的な関係にあり、システム開発においては、トップダウンによる設計とボトムアップによる実装・統合のバランスが求められる。

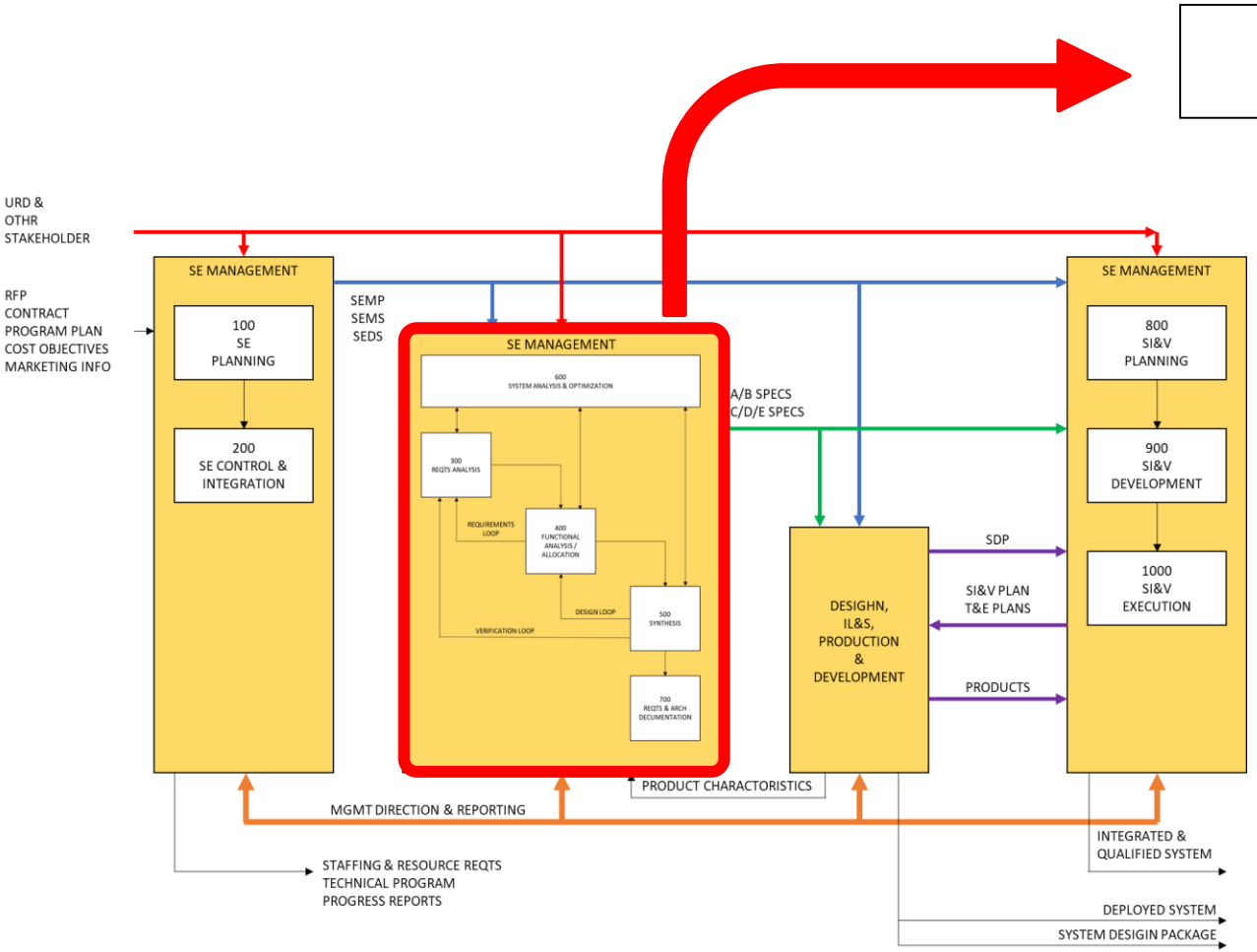
IEEE1220システムズエンジニアリングのプロセスの流れ

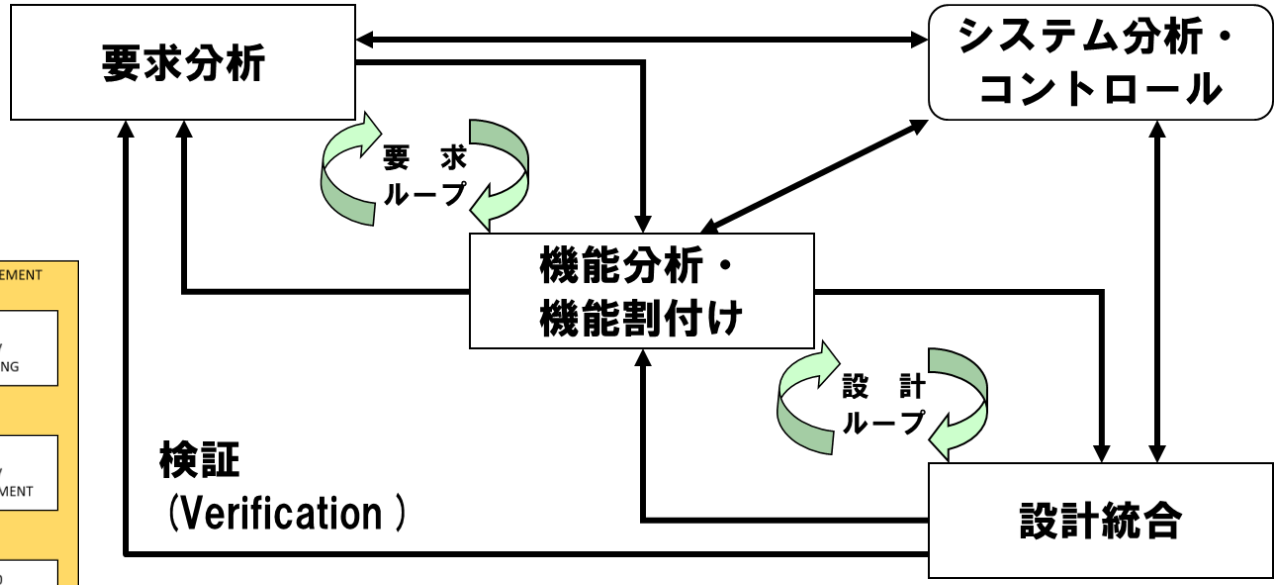


IEEE1220システムズエンジニアリングのプロセスの流れ

ID	プロセス	記述
1	要求の分析	システムに対するニーズや関心（利害関係者要求）とプロジェクトや事業上の制約，外部からの制約などの矛盾を考慮し運用シナリオと効果測定を定義して，これらの制約を含むさまざまな要求のトレードオフ分析と評価を行う。これにより開発中のシステムに対して要求の基準となる，いわゆるシステム要求を得る。このシステム要求は曖昧ではなく，計測やテストが実施可能でなければならない。
2	要求の妥当性確認	システム要求について妥当性を確認する。利害関係者の期待との比較，プロジェクトや事業上の制約，外部からの制約との比較により，相違及び矛盾を特定し，妥当性を確認したシステム要求を確立する。
3	機能の分析	妥当性を確認したシステム要求に沿って，システムレベルでの機能のコンテキスト分析を行う。これにより，外部システムに対する機能的振る舞いを洗い出す。そして機能間のインタフェースを定義し，性能要求を機能に割り当てる。
4	機能の検証	基準となるシステム要求に基づき機能アーキテクチャを検証する。アーキテクチャの完全性，機能及び性能の評価基準，制約の充足を検証し，重複，相違，矛盾を特定する。重複，相違，矛盾が存在する場合には，機能分析，要求分析まで戻ることになる。
5	総合	機能のグループ化，物理への割り当てを行い，機能アーキテクチャを満たす設計解を特定する。幾つかの候補から，安全面や環境面，ライフサイクルでの品質要因，技術要求などの観点でトレードオフ分析と評価を行い，その設計解をシステムが提供できるように設計要素を配置した物理アーキテクチャを決定する。
6	設計の検証	設計された物理アーキテクチャを検証するアプローチを選定し，それに基づいて，検査，分析，実証，及びテストの要求検証手順，検証環境を定義する。機能アーキテクチャの検証の場合と同様に，アーキテクチャの完全性，機能及び性能の評価基準，制約の充足を検証し，相違や矛盾を特定する。相違や矛盾が存在する場合には，機能分析，要求分析まで戻ることになる。物理アーキテクチャが検証されることによって，仕様と構成の基準が確立され，システムの分解構造が定まる。
7	統制	(1)から(6)のプロセスから得たデータ，設計解の構成，インタフェース，リスク，技術の進展を統制し，システムズエンジニアリングプロセスのリスクとアクティビティをやりくりする。

SE MANAGEMENT PROCESS





分解と統合のバランスを適切に取る

システム開発における分解とは、要求や機能をより細かい要素へと分割するプロセスである。

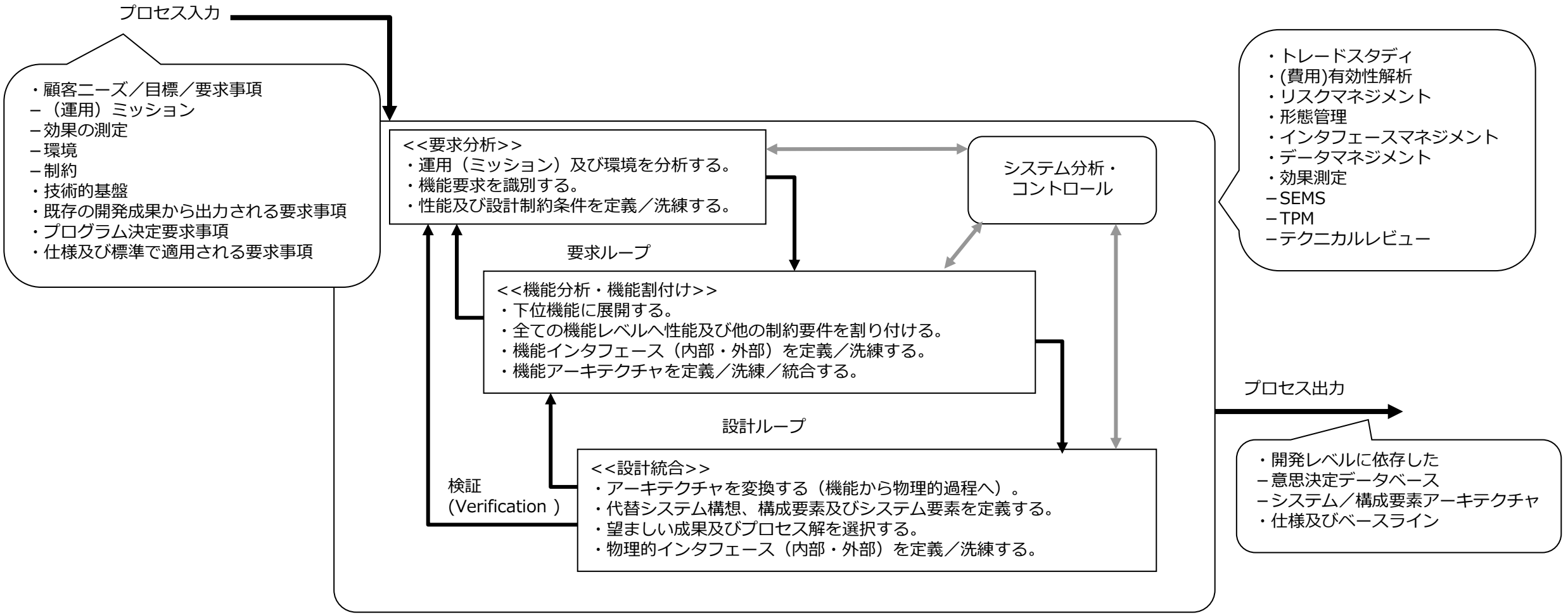
これは、複雑なシステムを理解し、設計を整理するための最も強力なツールの一つである。

しかし、分解の難しさは単に要素を分割することではなく、それらを統合し、全体として一貫性のあるシステムへと構築するプロセスにある。

この統合プロセスは、設計統合として示されており、機能や構成要素を結び付ける際の物理的、論理的適合性を考慮する必要がある。

- 要求分析 の段階では、利害関係者の期待や制約を抽出し、システムが満たすべき要件を明確にする。
- 機能分析・機能割付けの段階では、システムの要求をより基本的な機能へと分解し、それらの機能をサブシステムやコンポーネントに適切に割り当てる。
- 設計統合 の段階では、分割した機能を再統合し、システム全体としての整合性を確保する。このプロセスでは創発的な性質が現れ、システム全体の振る舞いを適切に評価することが求められる。
- システム分析・コントロール は、システム設計の全体を監視し、整合性を保つための重要なフィードバックループとして機能する。

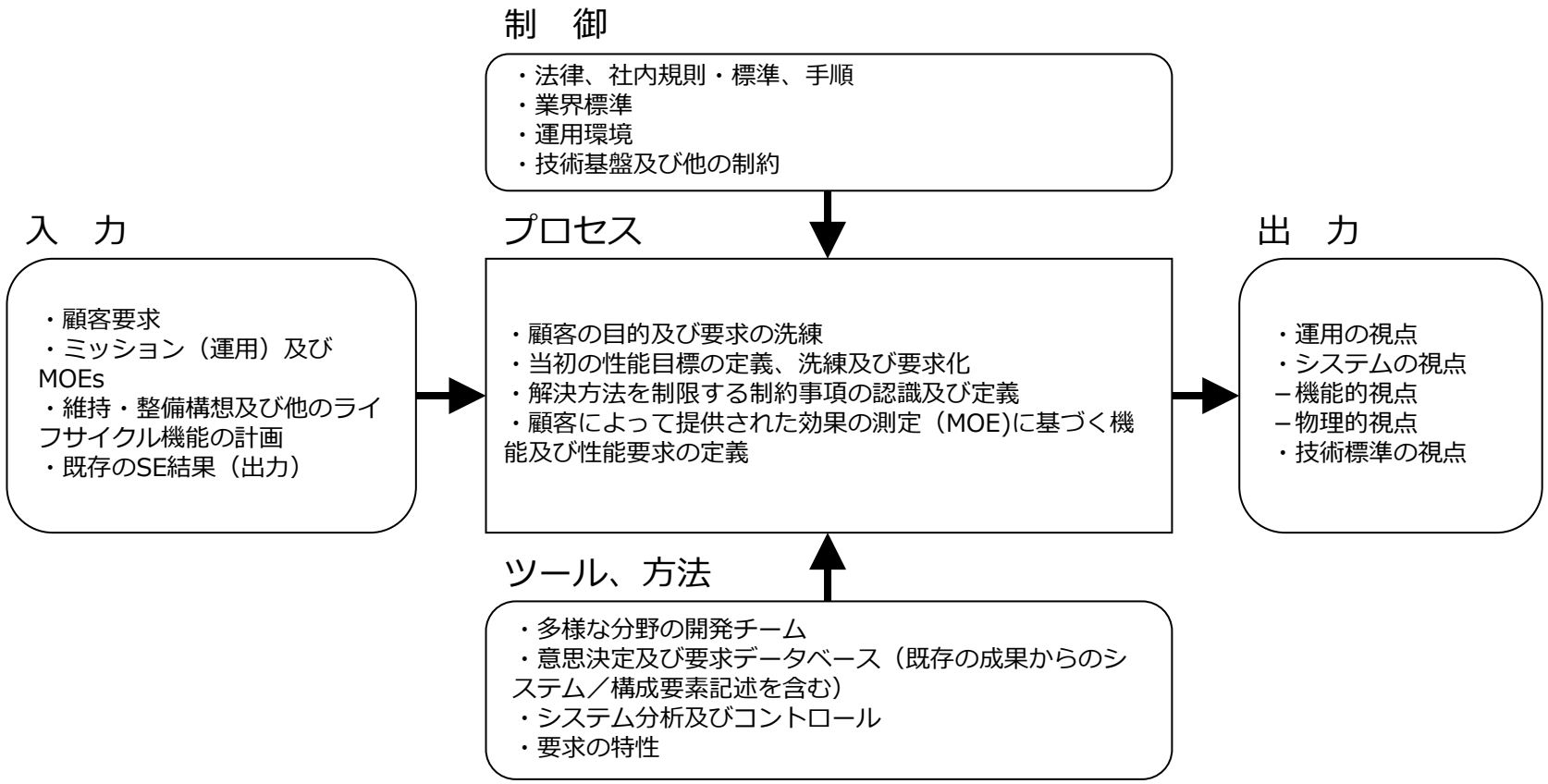
SE MANAGEMENT PROCESS



要求分析

要求

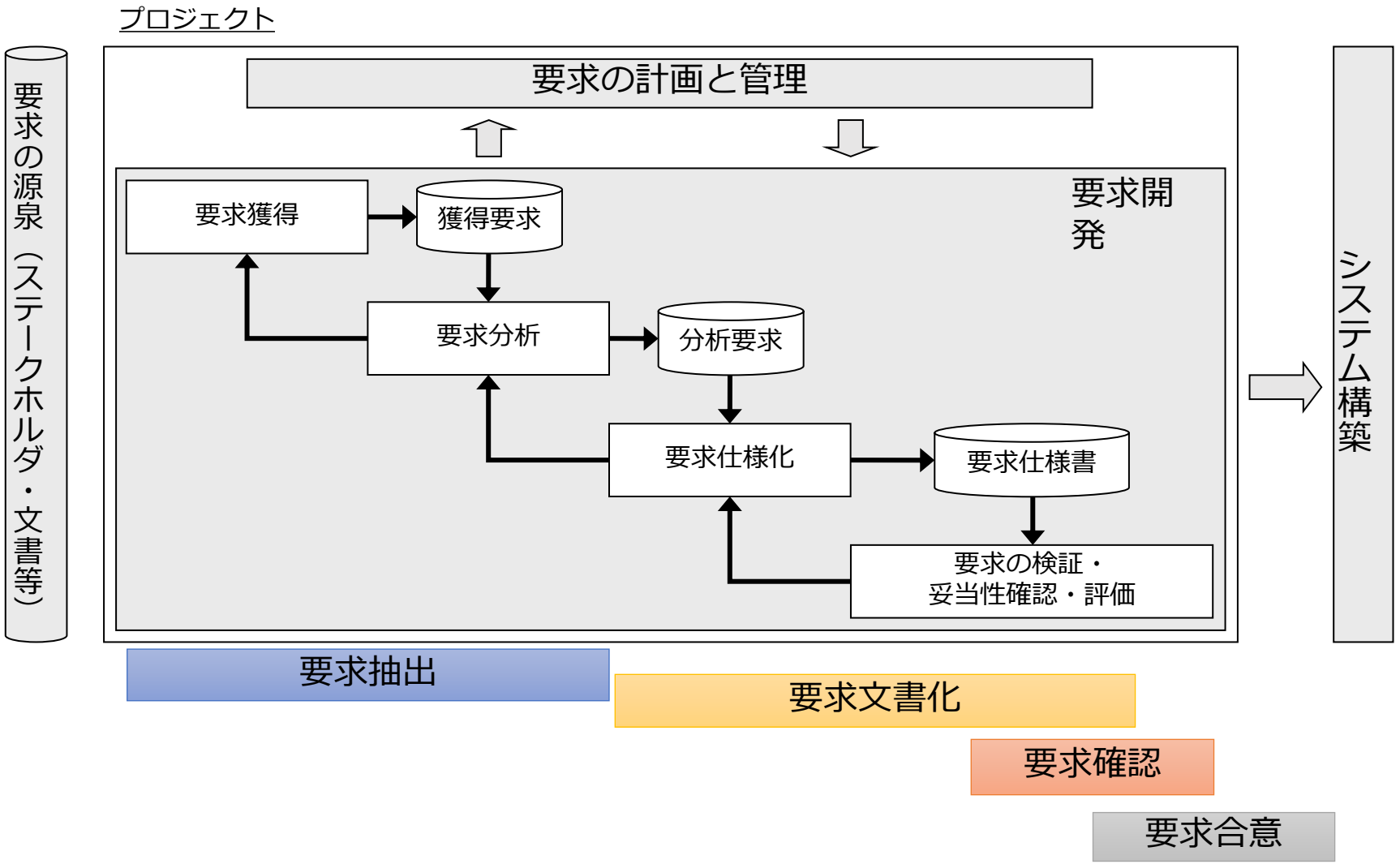
- 問題解決したり，目標達成するために，ステークホルダが必要とする条件，あるいは，能力
- 契約，標準，仕様，あるいは，その他の正式に要請された文書を満たすために，システム，あるいは，持つべき能力
- 上記の1，2の条件や能力を記述した文書



要求分析

- SEプロセスの最初のステップは、プロセスへの入力を分析することです。要求分析は、機能及び性能要求を明らかにするために行われます。
- 顧客要求は、システムが達成しなければならないことや、性能を示す要求の集まりに変換されます。
- システムエンジニアは、要求事項が理解可能であり、曖昧さがなく、包括的であり、完全性があり、かつ簡潔であることを確認しなければなりません。
- 要求分析は、機能要求及び設計制約事項を明確にし、定義しなければなりません。

要求開発プロセスと要求管理プロセス



システムのエンティティを明確に定義するために、システムの境界を適切に設定する

システムの境界を定義することは、何がシステムの内側に含まれ、何が外部に位置するのかを明確にする作業である。

特に 要求獲得や要求分析の段階では、ステークホルダーの期待を整理し、システムが果たすべき役割を決定するために、システムの境界が重要な要素となる。

システムを分析する際、我々は常にシステムを有限な範囲の中で捉える。これは、無制限に広がるエンティティの集合を考慮することは現実的でなく、また開発上の有益性も低いためである。

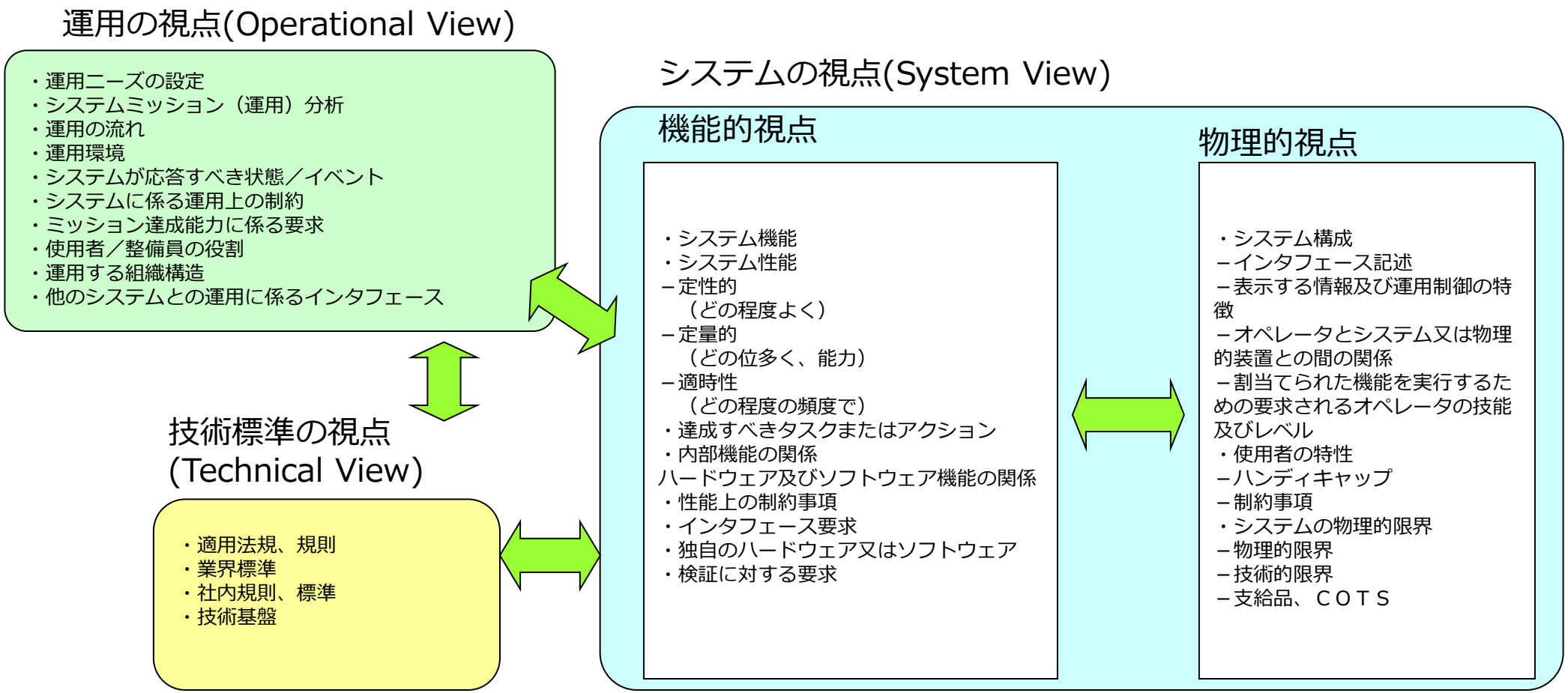
そのため、要求仕様化の過程では、システムがどこまでを対象とするのか、そして外部環境（コンテクスト）との関係を明確にすることが求められる。

システム境界の定義には以下の側面が含まれる。

- システム内部（要件の適用範囲）：システムが実装すべき機能や構造を含む。
- システム外部（コンテクスト）：システムには含まれないが、システムと相互作用するステークホルダーや他のシステム、環境条件を指す。
- システムの境界（インターフェース）：内部と外部を分ける境界であり、インターフェースを通じて情報や制御がやり取りされる。

このシステム境界の定義は、要求の妥当性確認や評価（Verification & Validation）にも影響を与える。要求がシステム内部で適切に実現可能であるか、また外部との適切な整合性が確保されているかを検証することが、要求開発の最終段階で求められる。

要求分析の出力


運用の視点(Operational View)
・運用ニーズの設定
・システムミッション（運用）分析
・運用の流れ
・運用環境
・システムが応答すべき状態／イベント
・システムに係る運用上の制約
・ミッション達成能力に係る要求
・使用者／整備員の役割
・運用する組織構造
・他のシステムとの運用に係るインタフェース
システムの視点(System View)
機能的視点
・システム機能
・システム性能
－定性的
（どの程度よく）
－定量的
（どの位多く、能力）
－適時性
（どの程度の頻度で）
・達成すべきタスクまたはアクション
・内部機能の関係
ハードウェア及びソフトウェア機能の関係
・性能上の制約事項
・インタフェース要求
・独自のハードウェア又はソフトウェア
・検証に対する要求
物理的視点
・システム構成
－インタフェース記述
－表示する情報及び運用制御の特徴
－オペレータとシステム又は物理的装置との間の関係
－割当てられた機能を実行するための要求されるオペレータの技能及びレベル
・使用者の特性
－ハンディキャップ
－制約事項
・システムの物理的限界
－物理的限界
－技術的限界
－支給品、ＣＯＴＳ
技術標準の視点
(Technical View)
・適用法規、規則
・業界標準
・社内規則、標準
・技術基盤

複雑なシステムのアーキテクチャを表現する際、個々の人間が直感的に理解できる情報量をはるかに超える膨大なデータが含まれている。
この情報をどのように整理し、提供すべきだろうか。

統合モデルと機能の投影の保持

システムの運用ニーズや制約、適用される規範といった情報を統合し、一元的なモデルとして保持する。

この統合モデルから、必要に応じて機能的視点（システム機能や性能、タスク達成の観点）や物理的視点（システム構成、オペレータとの関係、物理的制約の観点）を投影して可視化する。

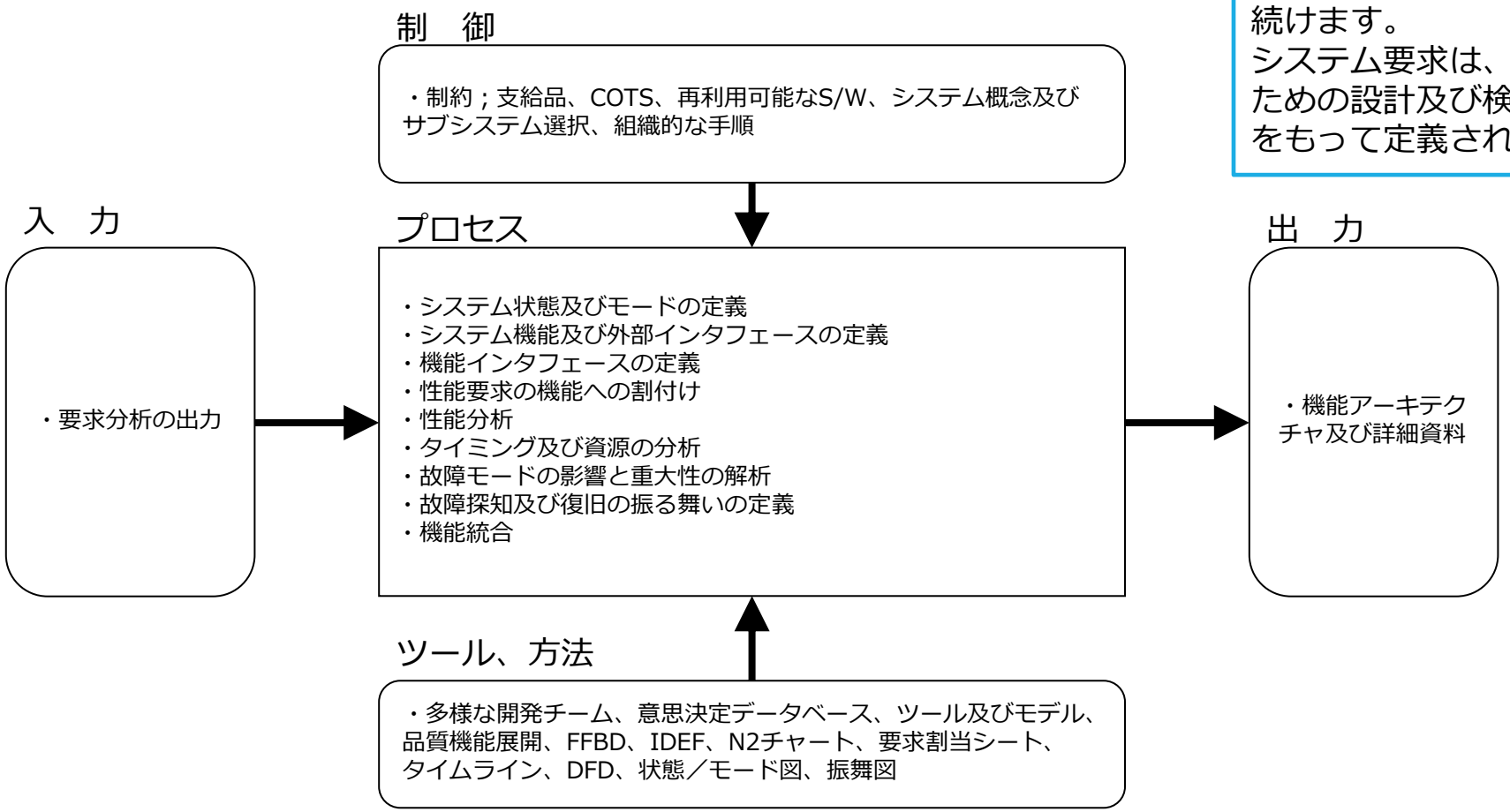
モデルに複数のビューを保持する

統合されたモデルを直接管理するのではなく、異なる観点（機能的視点・物理的視点）ごとに整理された複数のビューを維持し、それぞれの関係性を明確にする。これにより、システムの運用、機能、物理構造の情報を適切に切り分けながら、全体の整合性を確保する。

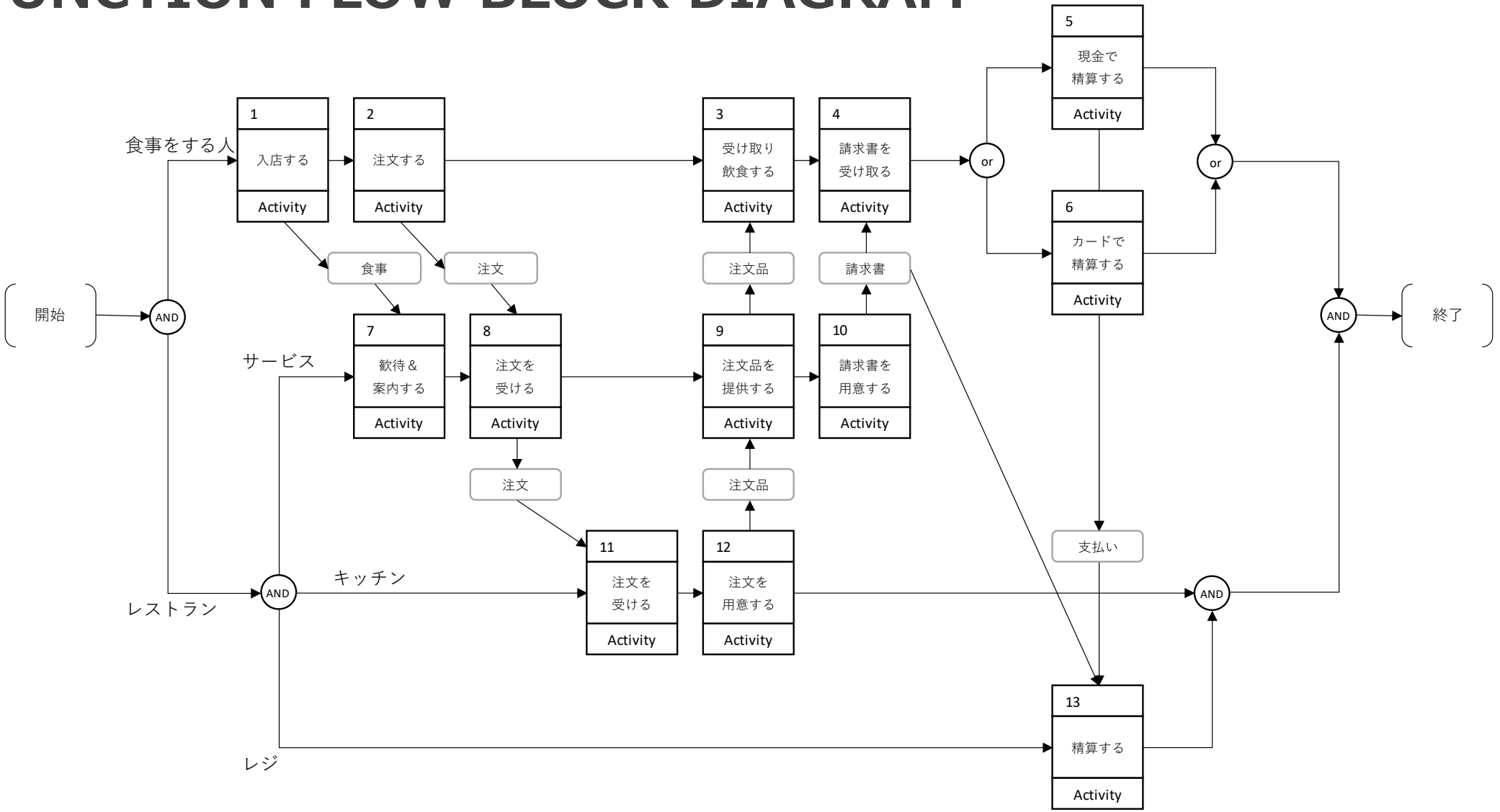
システムアーキテクチャにおいては、これらの手法を組み合わせることで、全体の統合モデルを維持しつつ、必要に応じて機能的および物理的なビューを取得することが可能となる。結果として、関係者間での情報共有が円滑になり、設計の整合性と適応性が向上する。

機能分析･機能割付け

機能分析・機能割付けのプロセス
システム内のいかなるレベルにある機能及び性能要求も高位の要求から開発されます。機能分析及び機能割付けは、下位レベルの機能及び性能要求に引き続いて、段階的に詳細化されるアーキテクチャ定義のように、繰り返し定義し続けます。
システム要求は、統合されたシステム設計をサポートするための設計及び検証基準を提供するために、十分な詳細度をもって定義され、割り付けられます。



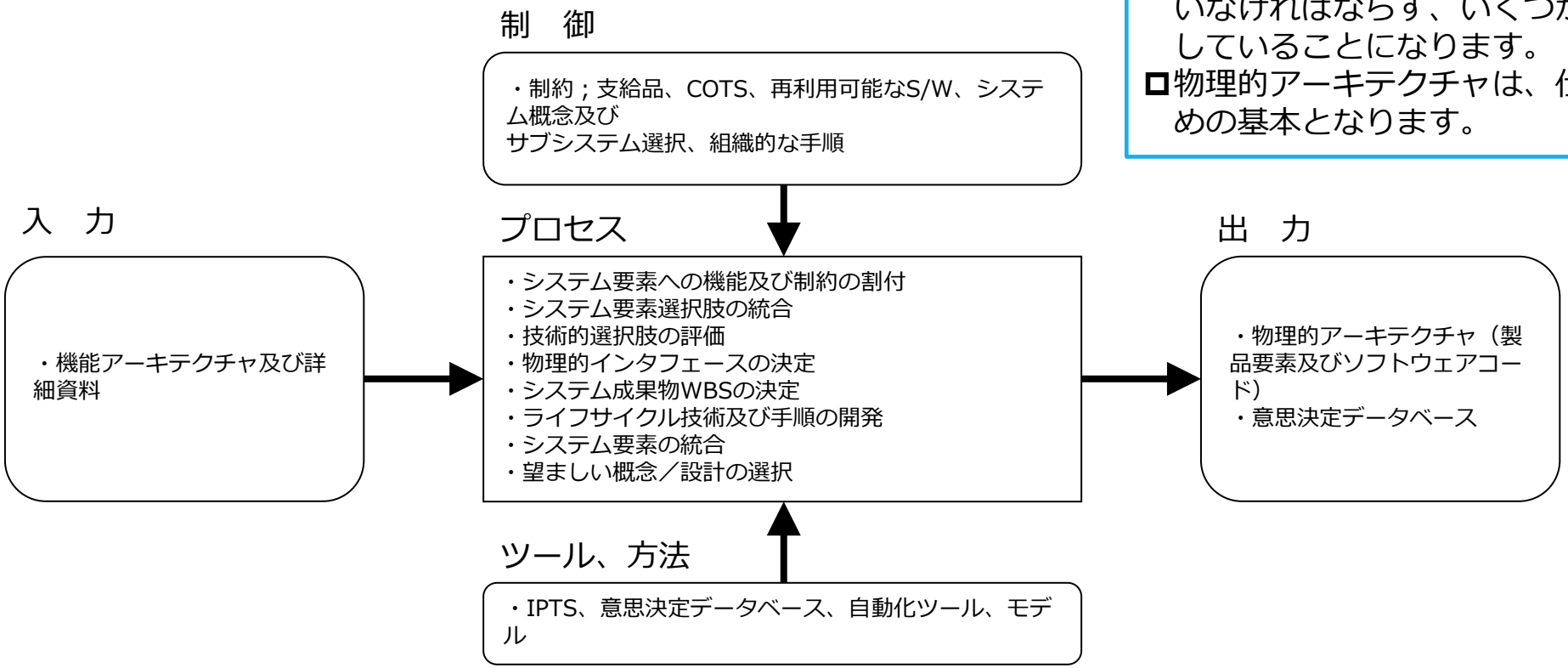
FUNCTION FLOW BLOCK DIAGRAM



設計統合

設計統合

- 設計統合は、互いに成立し、細目を定義するハードウェア及びソフトウェア要素に関連する成果物あるいは細目を定義するプロセスです。その結果は、物理的アーキテクチャといわれます。
- それぞれの部分は、少なくとも一つの機能要求を満足していなければならず、いくつかの部分は、多くの機能を満足していることになります。
- 物理的アーキテクチャは、仕様及びベースラインを作るための基本となります。



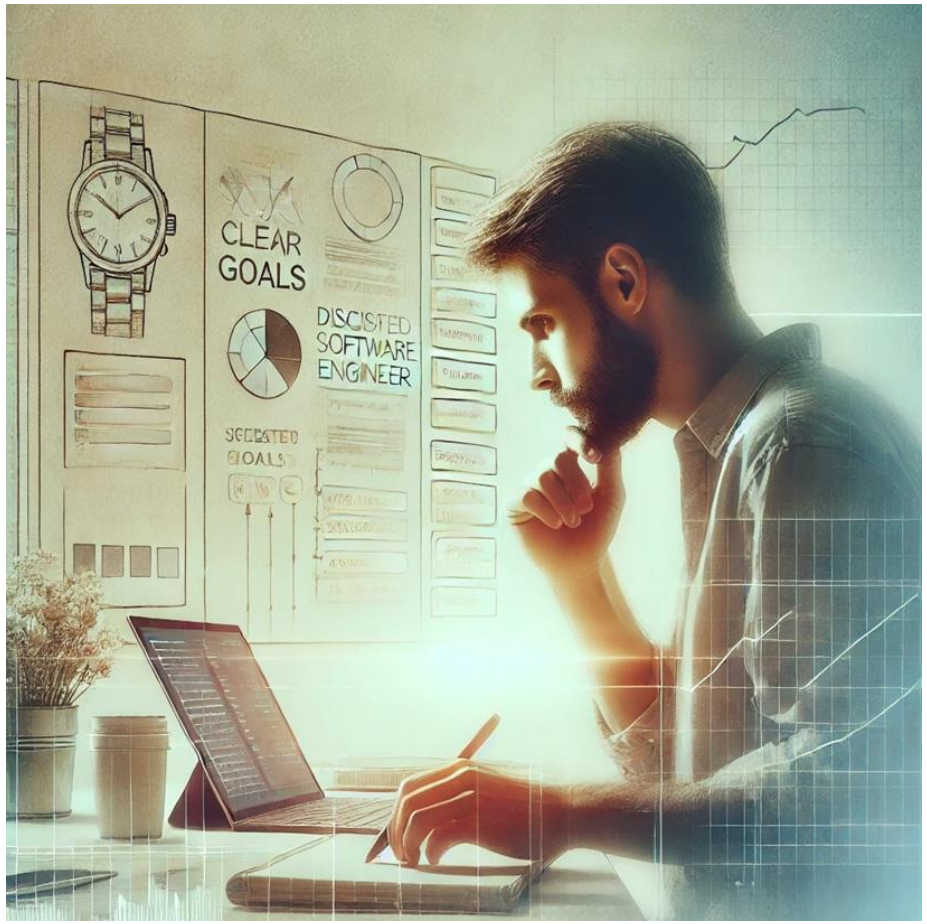
目標「規律あるSE」になるために

1. 規律あるSEになるためには、明確な目標を設定し、それを達成するための継続的な努力を意識的に行うことが不可欠である。
2. 目標達成のプロセスでは、計画的な実行、適切なフィードバックの活用、環境の整備、意識的な思考の実践 が求められる。
3. 目標に向かって自らの成長を管理し、規律ある行動を積み重ねることで、SEとしての専門性を高め、組織やプロジェクトにおいて真に価値ある貢献ができるようになる。

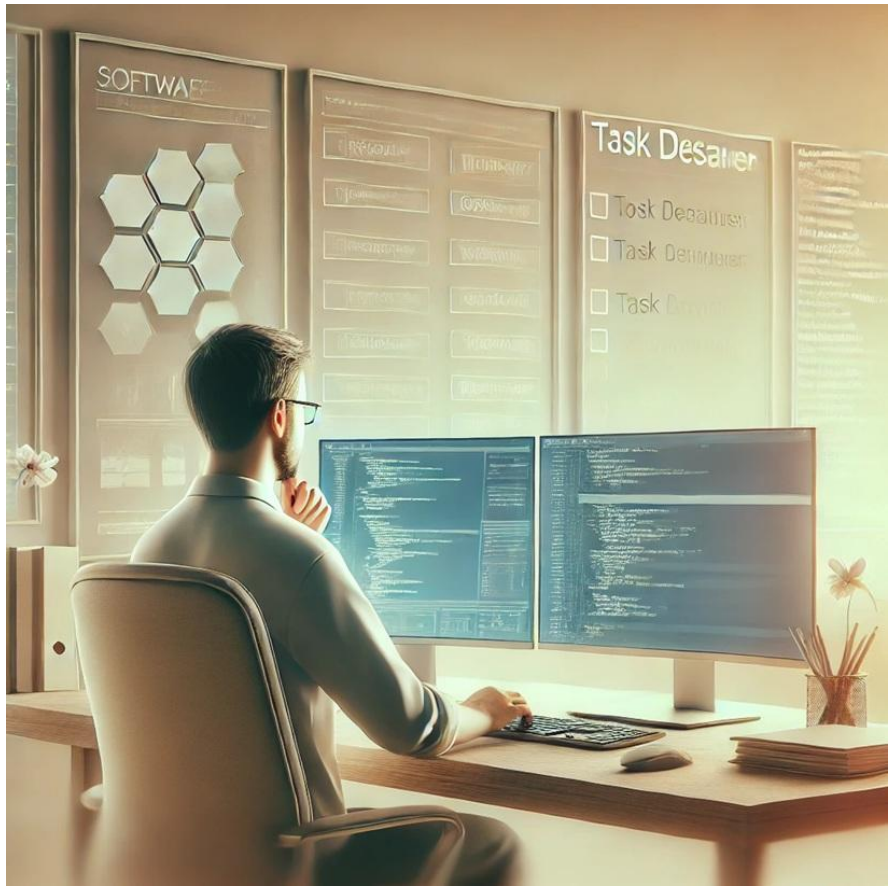
1. 規律ある目標設定

規律あるSEになるためには、目標を明確に定め、それに意識的にコミットすることが不可欠である。

目標は漠然としたものではなく、具体的で測定可能であるべきだ。例えば、「システムアーキテクチャ設計の専門知識を1年で習得する」「モデルベースシステムズエンジニアリング（MBSE）の手法をプロジェクトに適用する」といった、行動可能な目標を設定することで、規律をもって取り組む姿勢が生まれる。



2.規律ある実行



目標を達成するためには、何を継続し、何を新たに開始し、何をやめるべきか を明確にすることが重要である。

- 継続すべきこと: 効果的なSEの実践、定期的な振り返り、最新技術の学習
- 開始すべきこと: フィードバックを受け入れ、改善のための具体的な行動を計画
- 停止すべきこと: 非効率な作業、目的意識のない業務、惰性的なタスク処理

さらに、適切なフィードバックを受ける環境を整えることで、客観的に自分の進捗を評価し、調整することができる。チーム内でのレビューや、メンターからのアドバイスを積極的に活用し、軌道修正を図ることが重要である。

３.規律ある環境の整備

SEとしての成長には、状況的な制約を理解し、それを乗り越えるためのリソースを確保することも必要だ。

- 必要なツールや情報へのアクセスを確保する
- 時間管理を徹底し、継続的な学習を習慣化する
- 周囲の支援を得るためのコミュニケーションを積極的に行う

こうした環境を整えることで、計画的に学習と実践を続けることができ、SEとしての成長を加速させる。

４.規律ある思考法

目標を達成するためには、次の3つの思考プロセスを組み合わせることが重要である。

- 意識的な目標の選択 – 自分が目指す方向を明確にし、なぜそれを達成したいのかを理解する。
- 戦略と計画の策定 – 目標を達成するためのステップを具体的に考え、長期的な視点で行動を計画する。
- 努力の持続 – 一度決めた目標に対して、途中で投げ出さず、継続的に努力を積み重ねる。

意識的に目標を追求するSEは、「なぜこの取り組みが重要なのか」「どのようにすれば効果的に達成できるのか」を理解している。一方で、無意識に行動している場合、自己成長の機会を見逃し、目標達成に向けた改善の機会を失ってしまう。

長期的な視点で計画することは、なぜ非常に難しいのか

1. 現在は現実であり、具体的である。
2. 現在に対処することには、注意とエネルギーを要する：長期的思考は、現在のうちに未来を計画する機会を意識的に設けることを必要とする。先を見通すことに取り組まなければならない。
3. 多くの人は、長期的思考に必要とされる技能を身獲得である：広い期間に焦点を当てる必要がある。予測だけではなく、認知的統合が必要である。現在と未来の行為の間、異なる未来の行為の間における因果関係の理解が含まれる。未来を心的に具体化（想像）することも必要である。
4. 長期的思考は、自動的ではないため、意図的な心的努力を必要とする。
5. 長期的思考は、現在との統合が必要なため、優先順位づけを要する：すべての行為は、可能性からの選択である。

出典：北大路書房発行　ガブリエル・エッティンゲン, ティムール・セヴィンサー 著　未来志向の心理学

最後に

あなたが成功するために重要なのは、
あなたとあなたのパフォーマンスではない。
重要なのは社会であり、社会があなたのパフォーマンスを
どう捉えるかである。

（出典：光文社発行　アルバート・ラズロ・バラバシ著　ザ・フォーミュラ