モデルベースシステムズエンジニアリングプロセス（V2 Professor®）で進めるシステム開発の効率化

Introduction of model-based systems engineering process (V2 Professor®) that promotes system development efficiently

藤原 啓一*
Keiichi Fujiwara

我々は、メカ・エレキ・ソフト複合システム開発の生産性と品質の向上を目的として、理想と考えられるモデルベースシステムズエンジニアリング（MBSE）プロセスを構築し実践している。そのプロセス構築では、設計論に関する科学的背景を踏まえつつ、具体的手順として現場が理解・応用しやすくなるよう努めた。

この体系的な設計手法を「V2 Professor」と称する。

本論文では、V2 Professor に関する仕組みと特徴を紹介する。

We construct and are practicing the ideal model-based systems engineering (MBSE) process for the purpose of improving the productivity and quality of mechanical／electric／software complex system development. In the process construction, we tried to make it easier for the site to understand and apply as a concrete procedure, based on the scientific background of the design theory.

This systematic design method is called “V2 Professor”.

This paper introduces the mechanism and features of V2 Professor.

1. まえがき

MBSE を実践すれば「システムエンジニアリングの設計に関して、今までよりも良くなるはずである」という思いで導入されるケースが増えている。しかし、「良いこと＝準自動設計」と勘違いしていたり、MBSE ＝ MBD と考え、自動プログラミングに期待する場面に遭遇することがある。ここでは、MBSE の最大の利点である「モデル粒度で設計を進めることでトレーサビリティが分かりやすくなり、設計品質が向上する」という点に絞り、その理由を説明する。

2. システム設計効率化を阻む原理的な壁

システム設計には、誰も避けて通れないプロセス上の原理的な壁が存在する。その幾つかを、最初に紹介しておく。

2.1　要件仕様と構想設計の絡み問題

要件仕様と構想設計は絡み合う。制御系システムでは、情報系システムよりも、更に仕様間の絡みが複雑になる。これは設計における原理的な問題である。制御系システムは、ソフトウェア単独の場合もあるし、メカ・エレキ・ソフト複合システムの場合もあるが、仕様間の絡みが複雑となる問題に変わりはない。

仕様は、「要求（実現してほしいこと）を満たすための具体的な振る舞いの記述、手段やパラメータ」であるが、初めて作るシステムは、システムの仕組みに関する知見がないため仕様記述ができない。したがって、一度、設計してみて、作るモノの仕組み（アーキテクチャ）の目途を得てから、仕様を正確に記述するしかない。

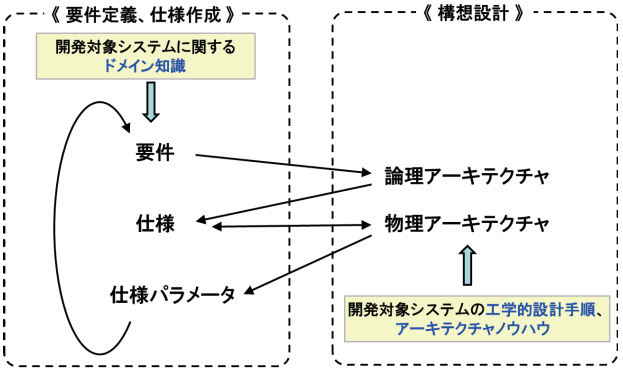


図１　要件仕様と構想設計の関係性

＊関西事業部

設計変更や不具合修正を、どのように各仕様に展開すれば良いのか（影響を考慮しながら修正箇所を探し出せば良いのか）という問題を解決するためには、最初の設計段階で、この仕様の絡み具合（仕様の関係性）を明らかにし、仕様の関係性を考慮した設計プロセス、設計手順を決めておくことが重要となる。

2.2　システム全体は部分の和以上である

「システムの上位レベルが働くためには、下位のレベルの要素そのものを支配する法則に依存する。しかし、上位のレベルの働きを、下位のレベルの法則によって明らかにすることはできない」（マイケル・ポランニー）。これは、上位のレベルの働きは、下位のレベルの働きの合計とは異なる部分が存在するということである。例えば、自動車の性能は、エンジン等の部品の性能に帰着するが、エンジンやタイヤの設計者を集めるだけでは、良い自動車は設計できない。自動車全体の働きを考えて、下位レベルの部品に性能割り付けができるエンジニアが必要なのである。つまり、縦割り組織のボトムアップ設計では、新規システムの設計は難しいということである。

このような問題は、スクラッチ開発が終わり、維持・派生開発を長らく実施した後に、スクラッチ開発を行うと必ずといって良いほど発生する。これは、維持開発段階になると効率化のために部品開発ごとに組織分けが行われ、「システム全体設計者がいない」状態で、創発的要素の多い新規システム開発を行おうとしたとき、システム全体に責任を負い、組織横断的な管理を行う能力のある人材を集めたチームをすぐに準備できないことが原因である。解決手段は一つ、システム設計に責任を持つ統括組織を作ることである。そして、大規模システム開発の場合には、適切な階層化設計プロセスに従って、上位層の適切な部分までを統括組織が責任を持って設計することである。

3. なぜ、MBSE は生産性を向上させる能力を持つのか

3.1　設計をモデル言語で表現する意義

MBSE は、システムズエンジニアリングの成果物をモデルで表現しましょう、と言っているにすぎず、そのベースとなるシステムズエンジニアリングを変えるものではない。これが理解できないと、MBSE を使うと、システムズエンジニアリングが上手くできるようになるといった誤解が生まれる。設計プロセスを背景としたモデル言語は、個々のモデルの中に他のモデルとトレーサビリティを取るべき要素が組み込まれているので、モデル検証でトレーサビリティ確保することを必須作業としてしまえば、モデルが完成された時点でトレーサビリティも確立された状態になっている。この点が「時間があったらトレーサビリティ設計を実施しよう」というようにトレーサビリティマトリクスを使って間接的にトレーサビリティを確保することと生産性において、格段の差を生む部分である。

3.2　制御仕様作成効率が上がる

制御系システムでは、仕様を表現するパラメータが多くなる。そして、そのパラメータどうしには関係性があり、2.1 節のとおり、作り方の分からない仕様は一度作ってみて、自分の作りたい仕様が定義できるだけの知識を得ておかなければならない。

仕様を作成するために行う、これまでの設計手順は図３のようになる。

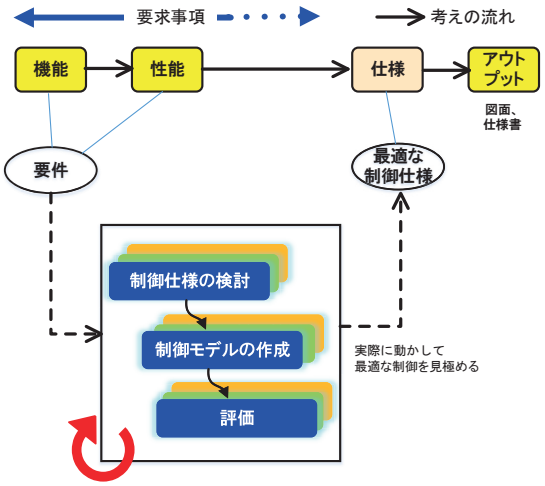


図２　制御仕様は設計してみて分かる

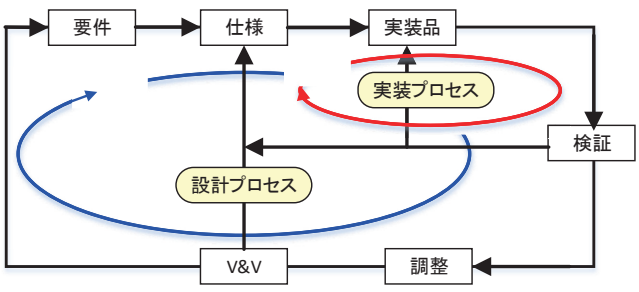


図３　これまでの設計手順

この手順の問題は、仮仕様を一旦決めて、その仕様と実装を整合させるために、実装プロセスを何度も繰り返さなければならないことである。また、検証段階で発見される設計ミスもあり、要件レベルからやり直すことも生じる。製造工程に渡せる仕様を決定することが設計の役割なので、プロジェクト全体の生産性向上はいかに早く仕様を決めるかにかかっている。

これを、図４のように、モデルベースの設計手順で行うと、仕様を決めるのに作ってみるという点は変わらないが、作って試すことを、モデル＆シミュレーションで行うので、実物を作るよりはコストを安くでき、コンピュータ上で繰り返し検討することが可能となるので試行錯誤が容易となり、工程も短くできる。

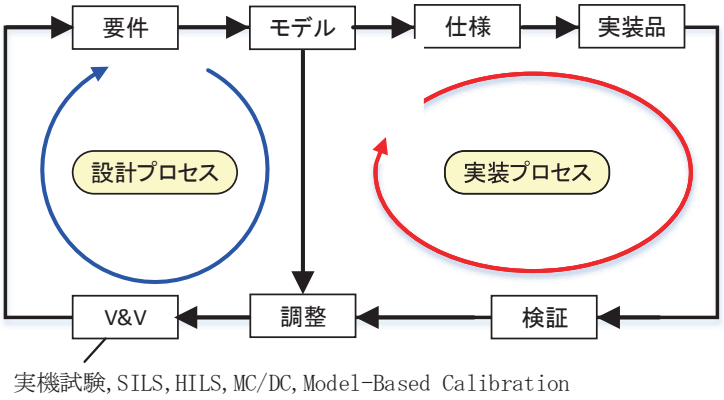


図４　モデルベースの設計手順

4.『設計プロセス』の設計に利用した参照アーキテクチャ規格

V2 Professor を構築するに際して、表１のような、設計プロセスに関係する規格を参考にしている。

表１　参照アーキテクチャ規格

規格	説明
エンタープライズアーキテクチャ（EA：enterprise architecture）	EA は、企業のミッション実現を目的として、必要な IT システムを最適に作り上げるために、主として発注者の視点でまとめた開発プロセス規格である（メタ・アーキテクチャ記述ともいう）。アーキテクチャ記述とは、製品のアーキテクチャを決めるためのプロセス規格であり、メタ・アーキテクチャ記述は、各企業固有の要求分析やアーキテクチャ設計のプロセスや手順を決めるための枠組みである。
DoDAF（DoD Architecture Framework）	DoDAF は、EA をベースにしながら、DoD（米国国防総省：United States Department of Defense）の戦闘運用及びビジネス運用・プロセスの両方におけるアーキテクチャ記述の開発、表現及び統合を定義した枠組み（フレームワーク）である。
IEEE 1220-2005	IEEE 1220 規格（2005 年）は、分析／統合（アナリシス／シンセシス）をどのように行うかを示したシステムズエンジニアリング標準である。ここで定義している分析／統合の考え方は、システム設計の全体プロセス、個別プロセスである企画プロセス、構想設計プロセス、実体設計プロセス（メカ、エレキ）等々、どこにでも当てはめることができる。

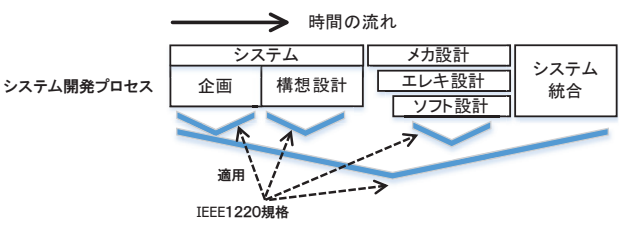


図５　アーキテクチャ規格のシステム開発プロセスへの適用部分

5. 実務で使える MBSE 言語が必要である

モデル言語である SysML や UML は、実プロジェクトで表現している設計モデルの要素的表現しか定義されていない。そこで、我々はコンセプト定義からソフトウェア開発までの設計モデルを表現するメタ言語 V2 Professor を作成した。V2 Professor は厳密な文法を定義することにとらわれず、モデル間のトレーサビリティを確保しやすくすることを目的にしたモデル言語であり、その特徴は、以下のとおりである。

(1)　“開発現場向け”モデル言語の意味

いまだ、設計書からの自動コード生成は主流ではなく、人手によるプログラミングに頼っている実態を勘案すると、様々な能力の人が理解し、使える設計モデル言語でなければならない。つまり、厳密ではあるが、それを読み解くのに数学的な技能が必須である形式言語や SysML を自由に駆使して表現したモデルでは、多くのレビューワーが理解するのに苦労し、総合的には生産性を落とすことになりかねない。そこで、モデル間のトレーサビリティ定義だけを厳密にしたモデル表現（言語）を定義した。

(2)　メタモデルとしてトレーサビリティを完備している

設計プロセスといわれる大雑把な粒度の開発フェーズを決めるだけに留まらず、プロダクトデザインに集中できるように、モデルレベルでの設計手順及びモデル間のトレーサビリティを示している。

(3)　開発方針

(a)　インタフェース設計重視

(b)　要件仕様とアーキテクチャ設計が分離できない点を許容できるプロセスの採用

(c)　要求・要件仕様の記述方法（レトリック）までのルールを完備

(d)　機能設計と機能展開の方法を中心に置き、システムの正常系設計と、異常系／安全系設計が分離しないようにする

(e)　設計ツールにおいて、V2 Professor を実現できない部分は自作する（例：用語の統一ツール）

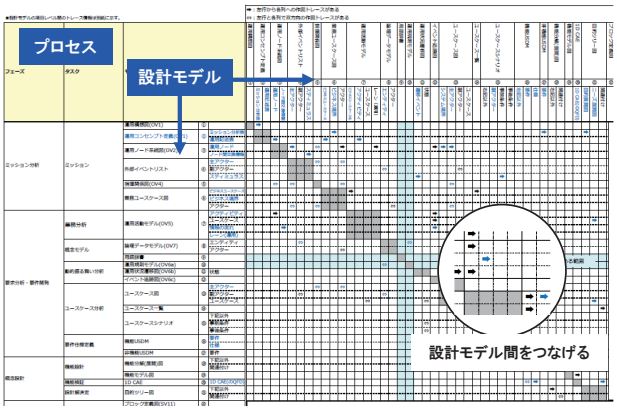


図６　V2 Professor のモデル間トレーサビリティの様子

5.1　モデル言語表現能力範囲

V2 Professor の設計表現範囲は、実装設計段階のメカ設計、エレキ設計を除く部分全てである。

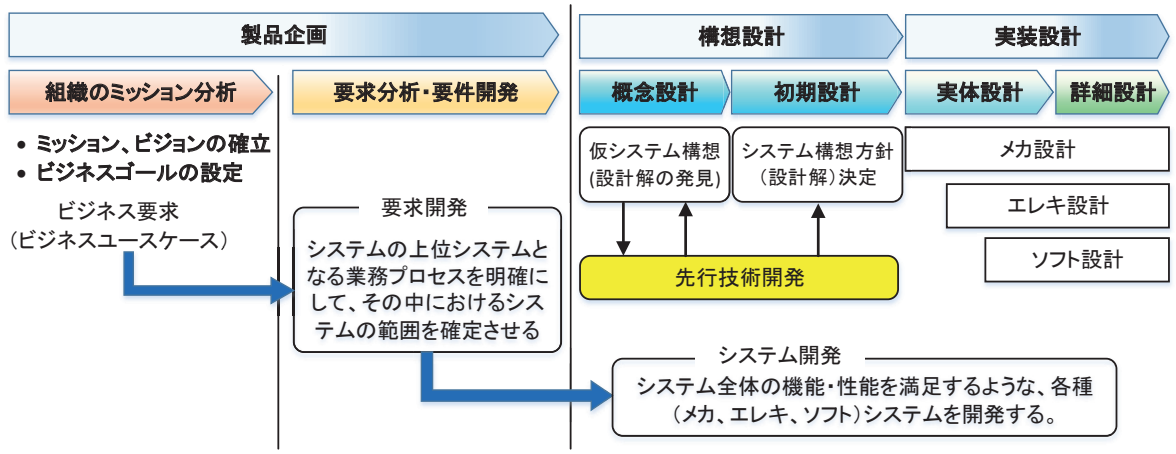


図７　システム設計プロセスの流れ

6. モデル設計プロセス／モデル作成・ガイドの概要

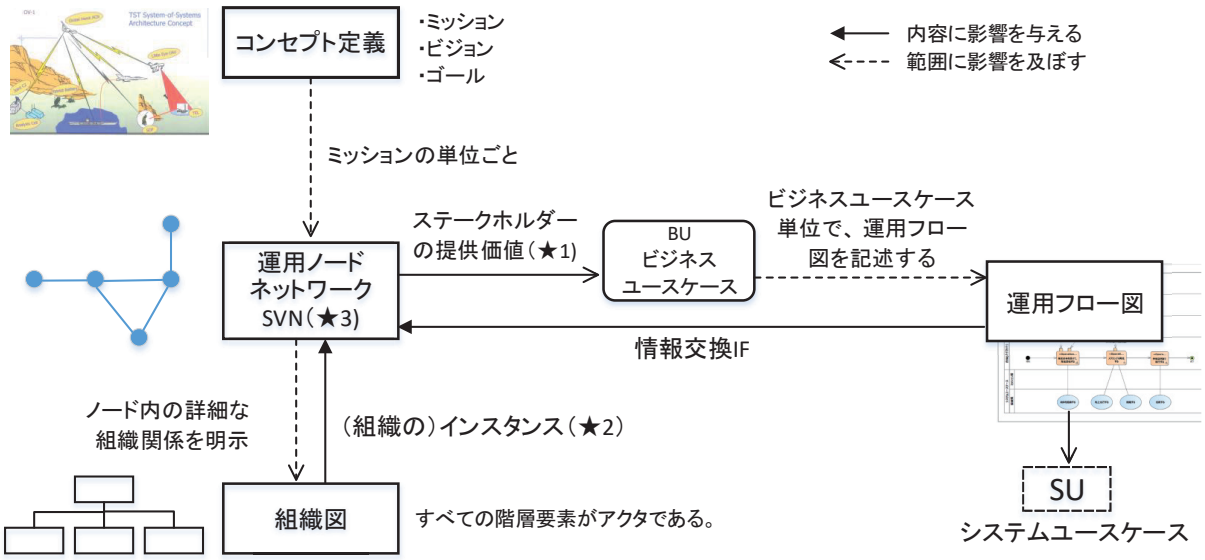
V2 Professor は、設計プロセスごとに適切なモデルを表現するために、モデル設計プロセスやモデル作成ガイド等を用意している。いずれもシステムズエンジニアリング設計知識と実践経験を積んだエンジニア向けのガイドである。

モデル作図ガイドは、特定のツール（例：Cameo Systems Modeler、Enterprise Architect、Genesys 等）でモデルを作成しようとするとき、ツールに用意されている記法の何をつかって記述すれば良いかを記載（規定）したものである。ほとんどの市販設計ツールには、V2 Professor のようなメタモデルは付随していないので、ほとんどのエンジニアは、何を、どのように使ってモデルを作成すれば良いのか迷い、自己の中で記述法を確立するまでにかなりの時間を要する。組織としての基準がなければ、各自の記述法が同一である保証はなく、レビューに時間を要するばかりか、見た目はトレーサビリティが取れているように見えるモデルであっても、コンピュータ上のモデルデータは、違うトレーサビリティが張られていることになり、検証に多大な時間を要する。

以下、モデル設計ガイド、作成ガイドで定義している内容の一部を紹介する。

6.1　ミッションからシステム機能要求までのプロセス

多くのシステム開発で、適切なシステムユースケースを抽出できない場合、運用フロー作成のプロセスを行うとシステムユースケースを抽出しやすくなる。組込システムといえども、製品コンセプトがあり、完全自動システムでない限り、製品を含めた運用があり（運用フローで表現）、それを明らかにすることで、システムユースケースが発見・定義できる。



★1:運用ノードネットワークのノード間には、あるノードからあるノードへ提供する価値を記述する。それが、ビジネスユースケースとなる。
★2:運用ノードは、組織をクラスと考えた時、そのインスタンスを意味するので、組織にないノード(アクタに相当)は、運用ノードとなって存在してはならない。
★3:個々のSVNを統合した(合体した)静的なSVNが存在した方が検証がしやすければ、それを作成する。

図8　コンセプト定義からシステムユースケースを導出するまでの関係性

6.2　ビジネス要求からシステム要件の導出

「あるビジネス要求の運用を表現する運用フローのアクティビティからシステムユースケースを導出する」ことを経験則として設計作業を行ってきている。これは、オントロジー工学の「行為は、機能と手段に分解できる」としていることと同意であり、経験則が間違いではないことを保証してくれるものと考える。

歩く（Operational Activity：行為）
→「場所移動する」（What：目的）
＋「両足交互運動方式」（How：達成方法）

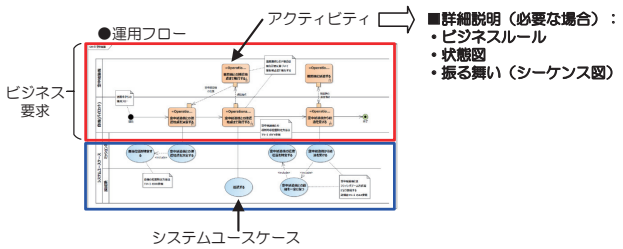


図9　運用フローの役割

6.3　システム構想設計

システム構想設計は、要件仕様作成と密接に関係していることは既に説明したとおりである。この関係性を具体的内容として明らかにしながら、階層的に設計していく様子を図 10 に示す。

6.4　モデルによる分析（アナリシス）と統合（シンセシス）手順

6.3 節で示した各階層の設計を、モデル粒度で進める概要手順を図 11 に、その詳細手順を、図 12 に示す。

システム構想（システムアーキテクチャ）設計とは、分析（アナリシス：analysis）と統合（シンセシス：synthesis）を実施して、複数の設計解原理を導いた後、設計解原理の組合せを評価し、全体の設計解を決定する作業である。ここで、分析とは、その段階のシステムの外部から見える機能（振る舞い）を定義する作業（論理方式設計）であり、統合とは、機能に対応する複数の設計解候補（代替案：alternative）を創出する作業（物理方式設計）である。

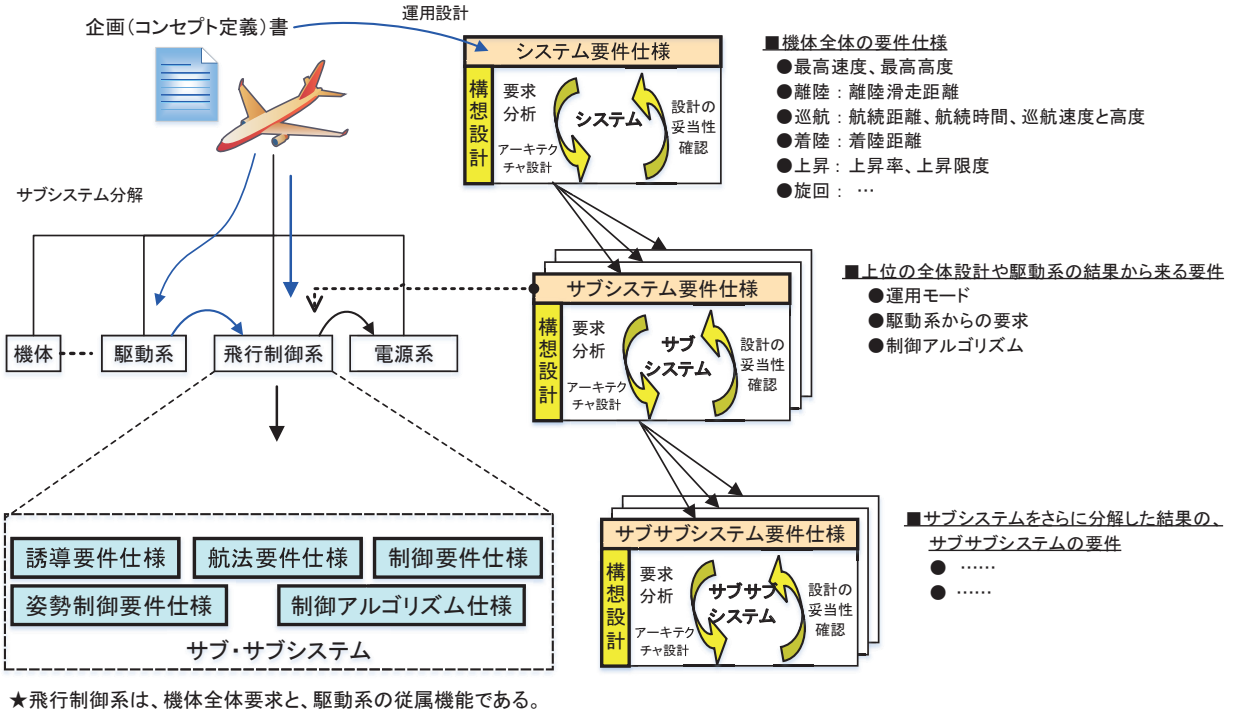


図 10　階層化設計の流れ

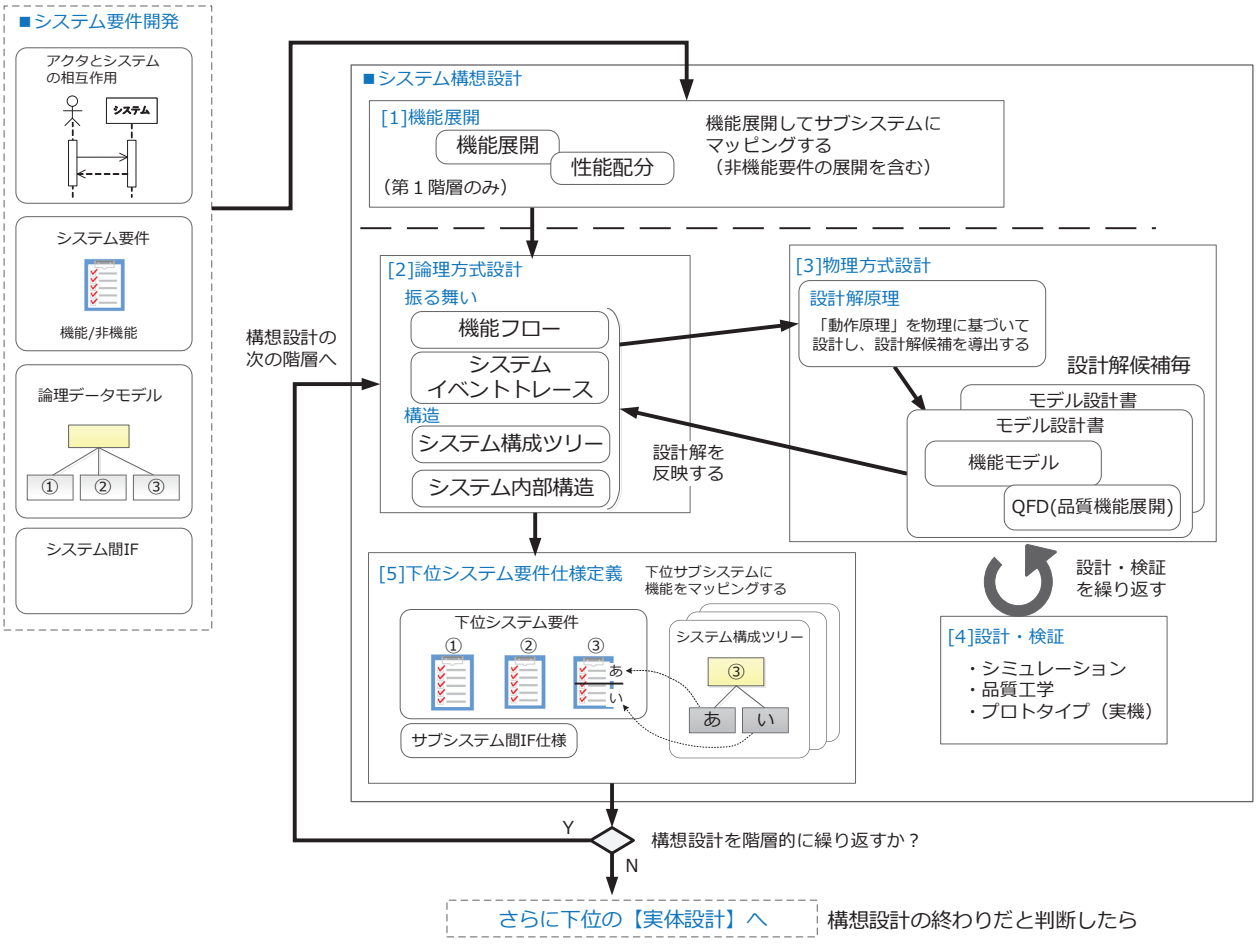


図 11　システム構想設計プロセス

■物理方式設計詳細

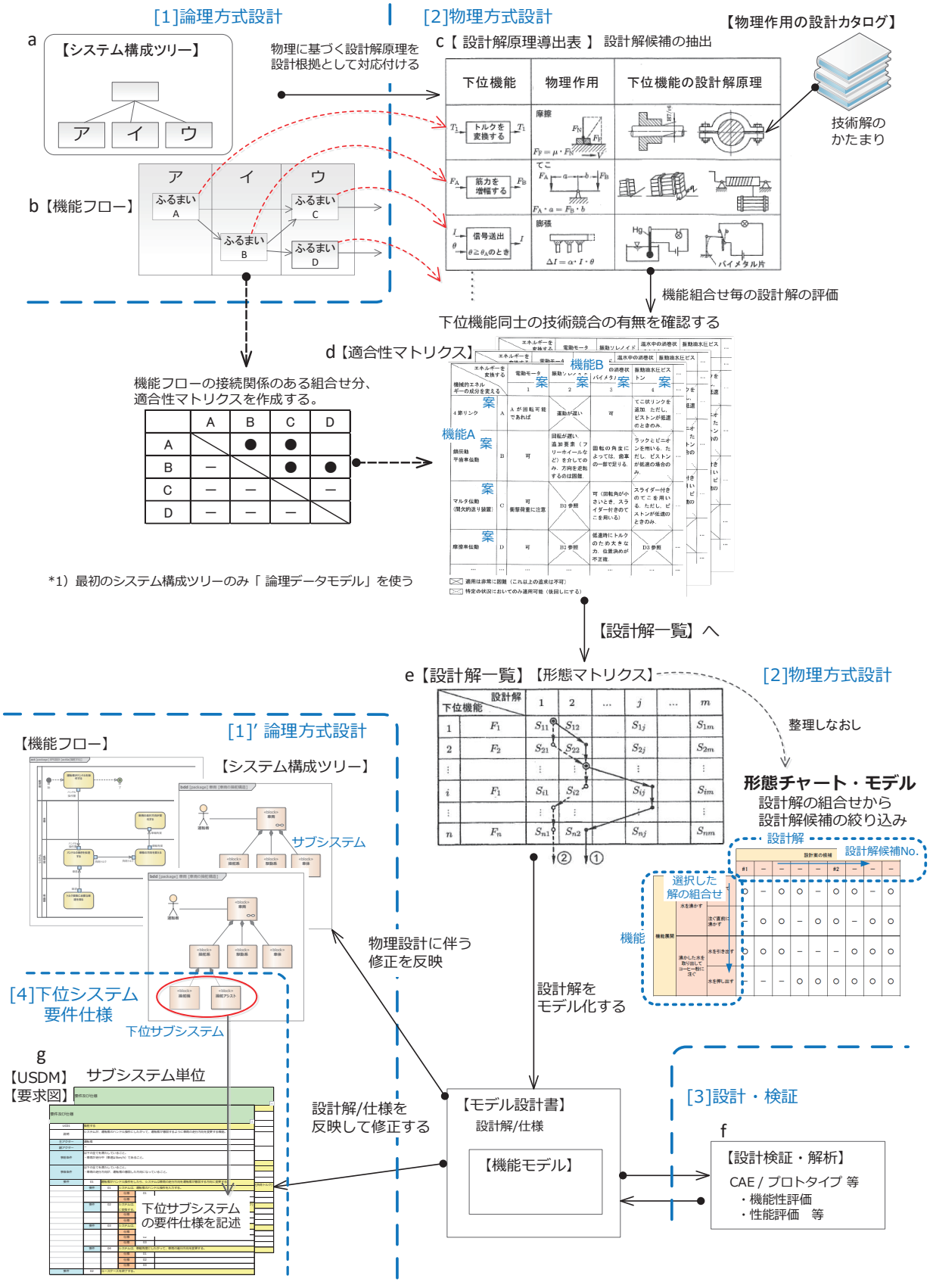


図 12　機能から設計解を導出するまでの手順

表２　システム構想設計手順内容の説明

手順		内容
a	一段下のシステム構成ツリーを作成する	設計対象システムから１つ下のサブシステムを定義する。サブシステムは概念モデルから作成した論理データモデルの中から選択する。
b	機能フローを作成し、「振る舞い」とサブシステムを関係づける	システム全体の振る舞いと整合が取れるように、機能フローの「振る舞い」をサブシステムに割り当て、サブシステム間のインタフェースを定義する。
c	設計解原理導出表を作成する	「振る舞い」を実現する適切な「物理作用」（物理原理）を考え、その物理作用に対応する設計解候補を考え出す。
d	適合性マトリクスを作成する	機能フローの接続関係のある組合せ分、適合性マトリクスを作成する。適合性マトリクス縦横には、接続関係にある「振る舞い」に対応する設計解候補を並べ、それらの組合せが、優先評価する非機能要件仕様（評価指標となる）に対して適切かどうかを全組合せ評価する。
e	設計解一覧を作成する	適合性マトリクスの全組合せの中から、有効な設計解を左から順に並べる。
f	モデル検証を行う	設計解一覧から有効性の高い順に検証を行う。検証方法は、そのフェーズにおいて適切なモデル検証方法（MILS ／ SILS ／ HILS ／ CILS）で行う。
g	下位のサブシステム要件仕様を作成する	サブシステムごとの要件仕様を作成する。
h	a ～ g を繰り返す	実装設計に移行できる段階でなければ、もう一段下位のサブシステム設計を行うために、手順 a ～ g を行う。

MILS（Model In the Loop Simulation）：プラント、コントローラ全てをシミュレーションモデルで構成
SILS（Software In the Loop Simulation）：MILS に対して、仮装 ECU 上に実際のソフトウェアを乗せて、シミュレーションループさせる構成
HILS（Hardware In the Loop Simulation）：SILS に対して、ソフトだけでなく ECU ハードも本物を使ったシミュレーション
CILS（Component In the Loop Simulation）：SILS に対して、自動車の AT なら本物の AT 本体（コンポーネント）を使ったシミュレーション

システム全体の要件仕様が作成されているとして、その後のシステム構想設計手順（図 12）の説明を、表２に示す。

6.5　機能展開（分析）

機能は、①挙動（自己の振る舞い）、②変換（入出力の変換）、③作用（外部のモノを動かす）の３種類に区分できる。ここではシステムの機能展開であるので、どの種別の機能においても、エネルギー／物質／信号が入出力となる。

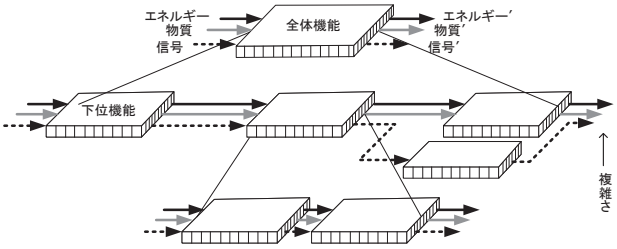


図 13　全体機能を下位機能に分解する

6.6　機能対応の構造を設計する

機能を実現する適切な「物理作用」と「物理原理」を考え、その物理原理に対応する設計解候補（機構：メカニズム）を考え出し、その様を設計解原理導出表に記載する。システムは物理原理を超えて動作させることはできないので、設計解候補とするモノの物理原理を明らかにすることが、設計の大枠の根拠となる。詳細な設計根拠は、適合性マトリクスの内容や、その内容を記載する根拠となる実験データや設計検討メモがそれに当たる。

表３　設計解原理導出表

	下位機能	物理作用	物理原理	設計解原理
記述内容	【機能】達成すべき目的	物理現象を支配する物理法則を、具体的な設計解を考慮せずに記述する。	物理作用と下位機能を結びつける働きを記述する。	【機構】具体化した方式

6.7　設計解の統合

メカ／エレキ／ソフト複合システムにおける論理設計と物理設計を繋げながら、システムを統合する考え方を図 14 に示す。

①機能展開を行い、②機能に対応する設計解候補を考え、③設計候補のうち、適切な組合せをシステムの設計解とする。④システムの設計解の妥当性を、シミュレーションや実験により確認する。⑤まだ、メカ・エレキ・ソフトに分けることができなければ（更に機能分解を行う必要があれば）、次の下位階層設計を行う。

このような一見回りくどい、余分な作業のように思える論理設計を実施するメリットは、以下のとおりである。

(1)　設計効率化：設計解原理（機構）の探索が容易になる。
　(a)　設計変更が容易になる。
　(b)　サブシステム間のインタフェース整合性が確認しやすい。
(2)　設計根拠の明示：論理設計は第三者に分かりやすい設計根拠資料の一つとなる。
(3)　結合試験時の不具合予防：構想設計の初期に機能モレが発見できる。

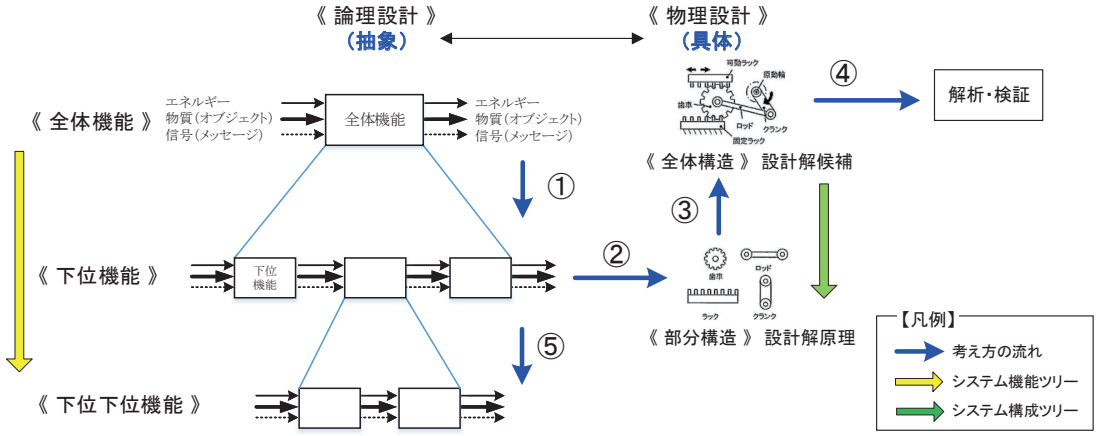


図 14　論理設計（機能）と物理設計（機構）の進め方

7. システムズエンジニアリングとしての設計ガイド

V2 Professor のシステムズエンジニアリングへの適用ガイド内容の幾つかを示す。これは、V2 Professor 自身の設計プロセスの一部でもある。V2 Professor は、システムズエンジニアリングの開発プロセスから必要とするモデルを一旦決定し、実プロジェクトへの適用からの差異を、モデルにフィードバックすることで、その表現方法を微調整している。

7.1　法律、業界基準（規定）のシステム開発への取り込み

世界で初めて作るモノでない限り、どの業界にも、安全な製品を開発するために最低限遵守しなければならない基準や運用時に守らなければならない法律が存在する（航空機の例：Certification Basis、Special Condition）。ところが、そのような基準や法律はドメイン内の幅広い製品に適応できるように作られているため、一読では理解できないほど一般表現されており、自身の製品に適用する場合は、解釈し、自社の製品開発が遵守していることを証明する必要がある。これを、システム開発エンジニアに解釈させるような開発プロセスにすると、無駄な作業、多大な工数を要することになる。そうしないためには、基準や法律を適切な開発プロセスフェーズに取り込み、エンジニア（最も貴重なリソース）を効率的に開発に集中させるために、法律や基準の解釈を多くのエンジニアが行わなくて済むように、「システム要件仕様」として落とし込み、エンジニアの理解できる言葉に変えておくことが重要である。

さらに、基準や法律を守っている設計プロセスと開発プロダクトになっていることをトレーサビリティとして社内外の第三者に証明する必要がある。V2 Professor では、分散保存された要求と設計の対応関係を、第三者に見せるために、IT の力を借りて、見やすい形に整形して作り上げるようにしている。

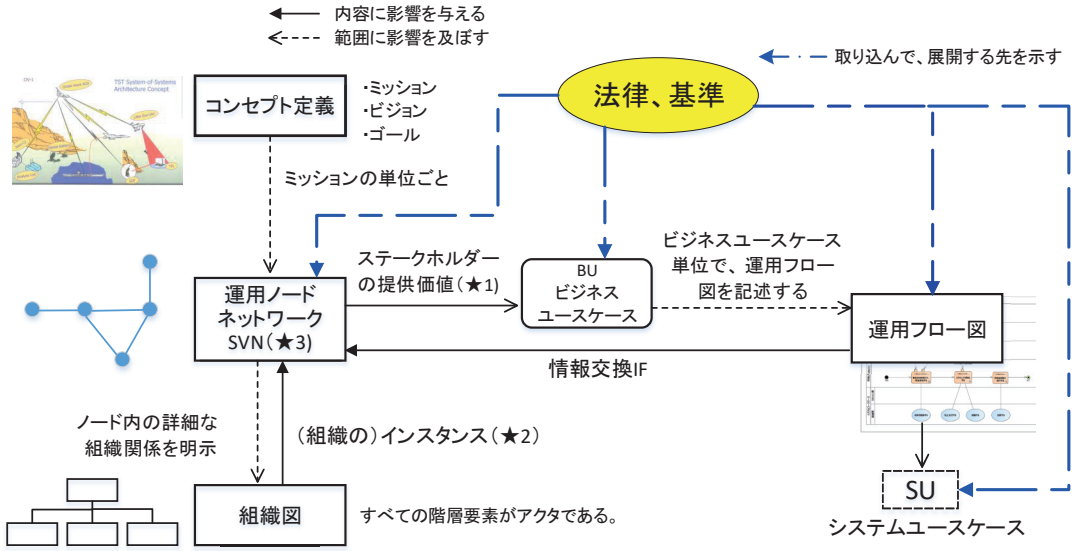


図 15　法律、基準は適切なプロセスに取り込む

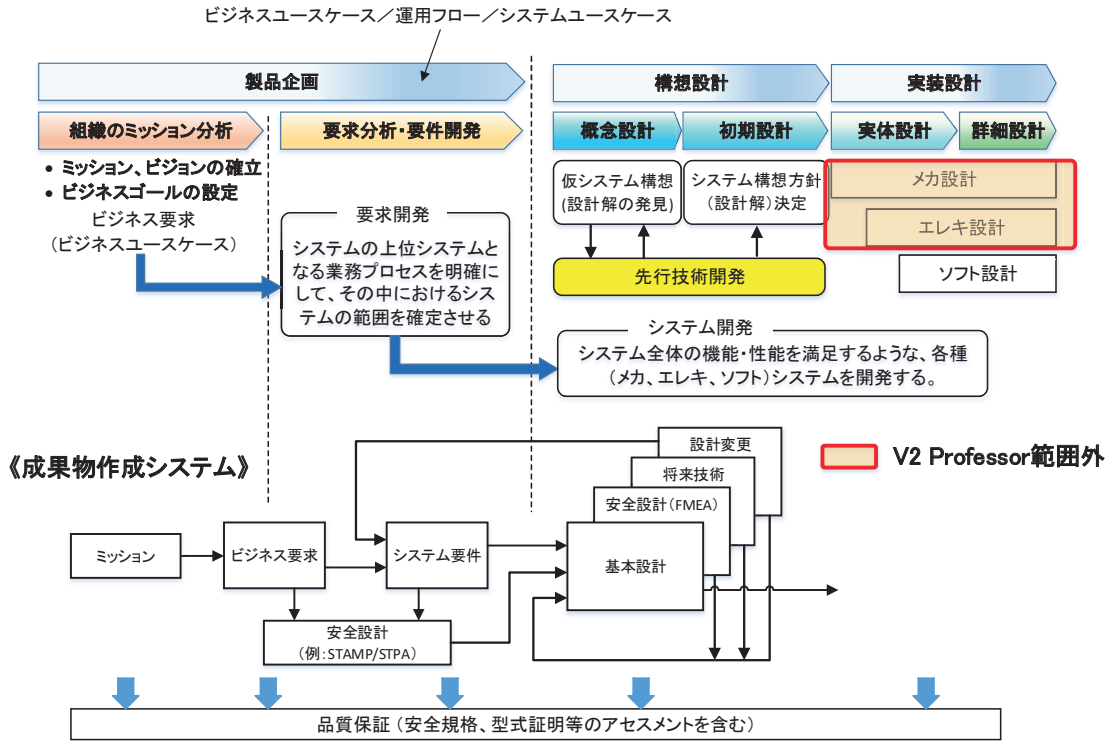


図 16　開発プロセスと IT 環境の関係

7.2　システム種別と要求定義種別の関係

V2 Professor では、要求や要件仕様を作成する形式として、仕様のヌケモレや矛盾を発見しやすい USDM（Universal Specification Describing Manner）を利用する。しかし、USDM は、要件に関係する仕様をすぐ近くに記載するため、要件を仕様に展開するための設計手法やプロセスを保持しない設計者が使うと、当該要件と整合性のない仕様を記述したり、他の要件、仕様と矛盾する仕様を記述してしまう可能性が高くなる。加えて、記述内容だけから、それらの問題を発見することが非常に難しくなる。

V2 Professor では、複雑な組込系開発の場合、要件から仕様を決定するまでの間には、必ず、構想設計（アーキテクチャ設計）の過程を挟むことを明示している。

また、開発対象システムの種類に応じて、ビジネスユースケースからシステム要件仕様までの成果物は、必要最小限となるよう、その選択指標を示している（表 4）。この中で、USDM の記述形式も、シナリオ法か、システム機能展開法かを選択するようにしている。

表 4　開発システムの種別と作成する要求・要件仕様モデルの対応関係

開発対象システム種別		ビジネス視点		システム視点			
		ビジネスユースケース	運用フロー	システムユースケース	システムユースケースシナリオ	システム要件仕様	
						シナリオ対応	システム機能展開
業務系システム	システム機能の主体は、人、ソフトウェア、外部システムであり、ハードウェアは、情報処理のコンピュータが主体であるシステム。	●	●	●	●	●	×
ソフトウェアシステム（カーナビ）	ソフトウェア・アプリケーションがシステム機能の主体であるシステム。要件作成方法は、業務系のミニマム版。	×	×	●	●	●	×
組込システム	いろいろな機器、機械に組み込まれて、その制御を行うコンピュータシステム。ソフトウェアは組み込まれる機器の制御をすることが目的である。	×	×	●	×	×	●
システム・オブ・システム（航空機）	メカが主流。システム機能は、メカ/エレキ/ソフトウェアのどれで実現するかが分かっておらず、構想設計で正確に分割することがシステム設計の主流となるシステム。	●	●	●	×	×	●

7.3　要求・要件仕様の設定項目フレームがノウハウである

制御系の要件仕様はアーキテクチャ依存度が高い。これは要件の項目がアーキテクチャとの関係性が高いということであるから、要件項目自体が制御系システム設計に関する知識依存度が高いということである。決めるべき要件のタイトルが項目リストとして事前に提供されるだけで、設計者は要件内容として何を決めれば良いか見当を付けることができ、かつ、記述内容を絞ることができるので、設計生産性を向上させる価値は非常に高い。

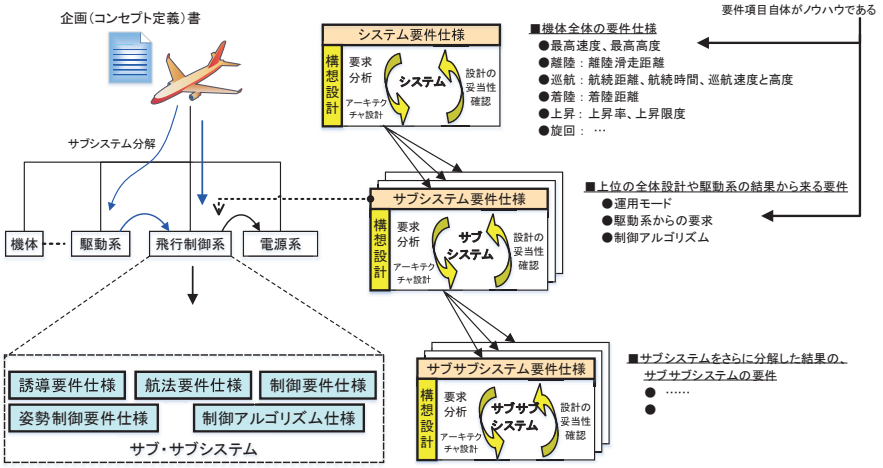


図 17　上位下位の階層間システムで要件項目は変わる

7.4　制御機能の階層化設計プロセスを中心に従属機能を階層化設計する

制御系システムにおける MBD では、制御器（コントローラ）の実装を睨んで、簡易モデルから詳細モデル、そして、実機検証モデルまで段階を追って詳細化しながら、モデルごとに徐々に仕様を決めていく。この一連のモデルの詳細化手順は、MBSE における階層化構想設計手順に沿って行う。MBD の制御器を設計する流れを図 18 に示す。ここで、プラントや制御器のモデルを検討するのに、設計解原理導出表の「物理作用」を考慮した設計解を利用する。

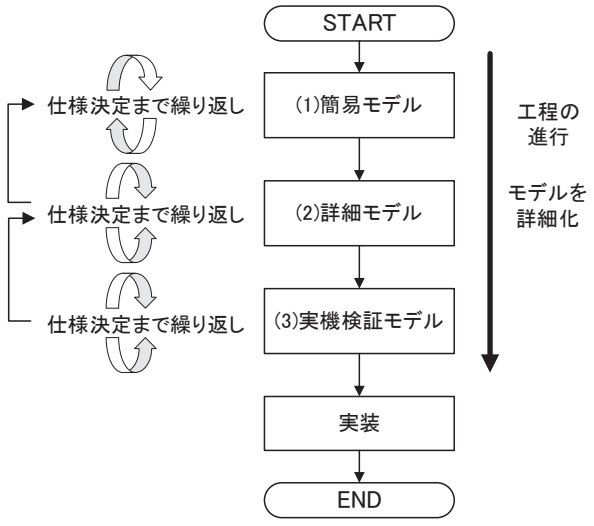


図 18　制御系システムの階層化設計フロー

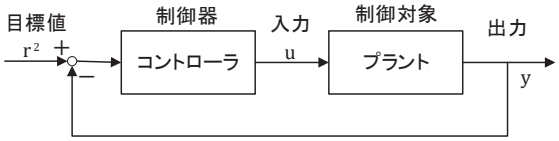


図 19　プラントと制御器（コントローラ）の関係

各レベルのモデルは、通常の制御系システム開発の手順で行う（表５）。

表５　制御系システム開発の手順

手順		設計内容
1	機能・機構の仮設計	当該設計対象階層レベルに対応する設計解原理導出表（物理原理）を利用して、モデル（センサ・アクチュエータ等）の機能と機構の仮定義を行う。
制御対象（プラント）のシステム同定		
2	モデリング	制御対象の物理モデリング、システムの同定を行う。
3	アナリシス	制御対象の性質の解析（安定性、減衰特性、定常特性など）
4	制御系仕様の決定	制御目的から導かれる制御系への要求を決定する。
制御器（コントローラ）の設計：古典制御、現代制御、ロバスト制御など		
5	制御器のゲインの算出	適切な設計法、PID 制御／ロバスト制御（H∞最適制御）などを選ぶ。
6	制御器の評価	数値シミュレーションや評価として妥当な簡易プラント実験を行い、設計した制御器の仕様の妥当性評価を行う。
7	制御器の実装、試験	実機に制御器を実装した試験を行う。

★１～７の手順をうまく行くまで繰り返す。

7.5　インタフェース設計

機能設計を行えば、必ず、機能間を結ぶ論理インタフェース設計を行わなければならない。階層設計を行う場合、上位／下位で整合のとれた論理インタフェース設計を行わなければならないし、最終段で明らかになる物理インタフェースとの対応関係も明確にする必要がある。

それらの整合を取るためには、サブシステムの階層化表現と同期しながら作成するインタフェース辞書を使って、インタフェースデータを階層表現することで実現する。これはデータフローダイアグラムの辞書を作成するのと同じ考え方である。インタフェース辞書を開発におけるモノを表現するモデルから独立させ、インタフェース定義データベースのように扱うことで、他の表現モデル（例：イベントトレースモデル）間のメッセージデータと整合しているかどうかも検証できるようになる。

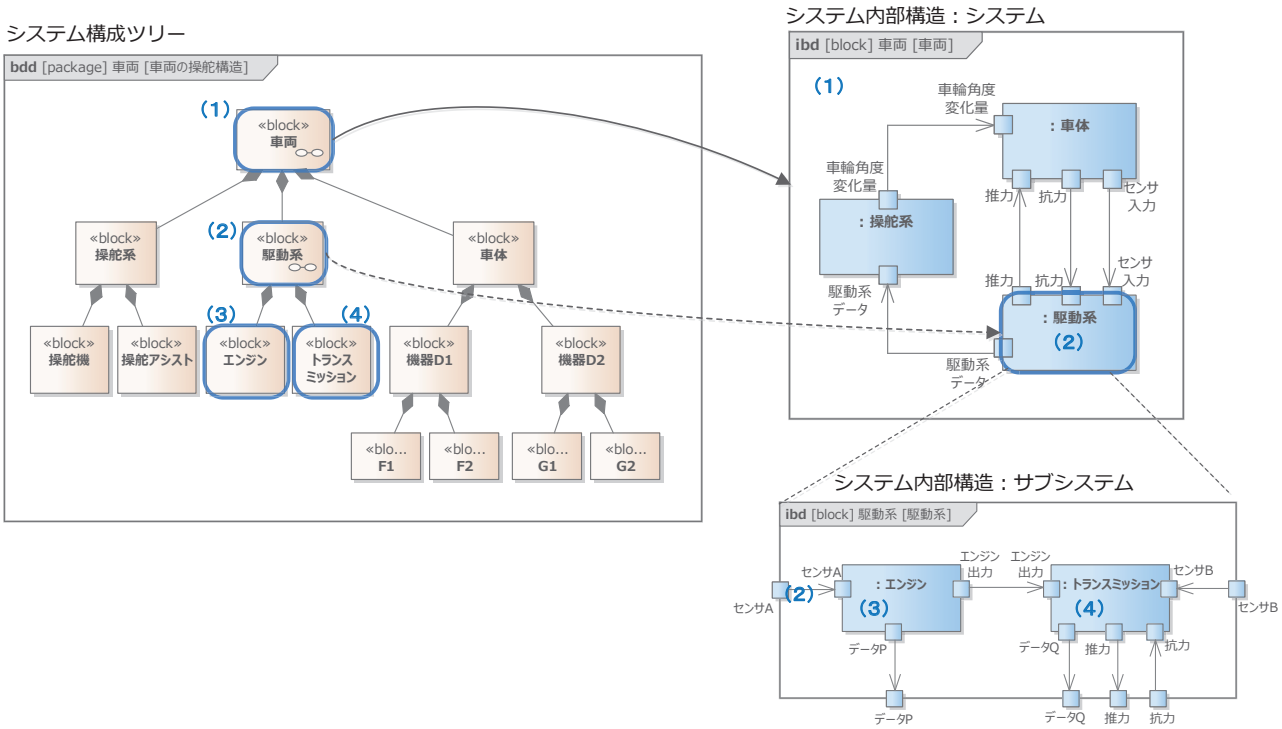


図 20　システム構成ツリーと対応するシステム機能フロー（システム内部構造）

インタフェース辞書

センサ入力 = センサA + センサB
駆動系データ = データP + データQ

車輪角度変化量
推力 = 前輪推力 + 後輪推力
抗力 = [摩擦力(静止時) | 摩擦力(走行時)]
エンジン出力

インタフェース辞書の凡例

記号	意味	説明
X=A	Aに等しい	データ名(左辺X)が、右辺に列記したデータ、値から成り立つことを示す。
A+B	A と B	複数の構成要素を1つのグループにまとめる。ただし、記載の順序に意味はない。
M { A } N	Aの反復	中かっこで囲んだ内容を任意回数繰り返す。繰り返し回数はM回以上、N回以下を表す。
[A | B]	A または B のどちらか	複数の項目から一つだけが選択されることを示す。
(A)	A の任意選択	必須ではない（なくてもよい）項目を示す。
/A/	構成要素	これ以上分割できないアトミックなデータ（構成要素）を示す。
/* A */	コメント	項目説明用の記述を示す。

図 21　インタフェース辞書

図 22　弊社の事業ドメイン

8. V2 Professor の実務適用能力が高い理由

8.1　なぜ、V2 Professor は、大規模システムや幅広い分野の設計に対応できるのか

弊社は、幅広い分野の組込系／情報系の開発を、システム開発支援からソフトウェア開発まで出荷後品質を保証する完全請負形態で行っており、開発で生じた本質的問題を把握できる立場にある。先に述べたように、その問題と対策は、V2 Professor にフィードバックし、改善したメタモデルを開発に適用することで実践検証を続けてきているので、規模の大小に関わらず、また、分野依存性の少ない開発プロセスのメタモデルを作ることができている。

8.2　なぜ、V2 Professor は、生産性と品質を向上させることができるか

V2 Professor は、弊社で実践している開発プロセスと品質管理・保証（CMM レベル４、Automotive SPICE レベル４）からの知見が反映されたガイドである。

弊社は、CMM、A-SPICE の理解活動を通じて、現場のエンジニアが自分たちの開発プロセスを主軸に品質規則を作り上げている。したがって、技術活動を支援する品質保証の仕組みが技術的な開発活動と齟齬無く動く仕組みができあがっており、品質保証活動だけが一人歩きし、現場活動とは異なる書類ができあがるということがない。さらに、V2 Professor の枠組み外ではあるが、現場の開発活動の問題点が、品質保証活動の中で、いち早く、第三者にも見える状態に保たれている。この品質保証（QC）の仕組みは、異常値を起こさない設計の支援活動であり、V2 Professor を外から支援する重要な役割を果たしている。

8.3　構成・変更管理システム

開発規模が大きくなればなるほど、設計変更の効率化が最も生産性に効いてくる。レビューをすれば必ず、一定量の設計変更・誤りが発見される。これを適切に設計に反映していくためには、リアルに設計書のトレーサビリティが確立されており、影響度を確実に把握できるよう制御されている状態が必要である。しかし、多くの会社では、後追いでトレーサビリティマトリクスを作成して…という開発とトレーサビリティの間に遅れが発生する状態が起きている。その間にも変更は生じるので、開発とトレーサビリティのズレが大きくなる。V2 Professor はモデル作成の段階でトレーサビリティを取っておき、レビュー後には、再度、トレーサビリティを確立し直せるというサイクルを設計行為の中で回すことができる。

9. V2 Professor を使った工程管理

9.1　品質プロセスと技術プロセスの関係

V 字プロセスは、時間軸を表していない。設計をアジャイルに実施するためには、いかに設計変更の反映可否を判断し、反映実行を完了させることができるかが鍵となる。設計書間のトレーサビリティが完備されているプロセスフレームがあれば、変更箇所が特定できれば、そこを中心に影響が及ぶところまで検討を広げて行きやすくなる。新規開発の場合も、開発しやすい箇所から始めれば良い。

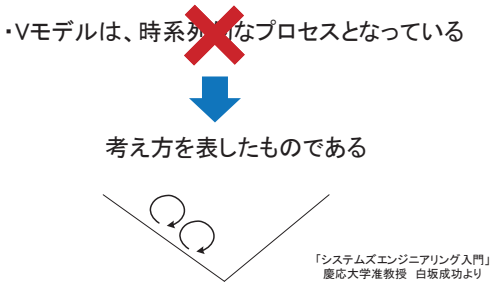


図 23　V 字モデルの意味

9.2　アジャイル管理のコツ

V 字モデルは設計の進め方順序を示すモノではなく考え方を示すものであるからこそ、推論できる人が行う派生開発やボトムアップ開発は成り立つ。現実の開発現場を見ると、新規設計管理のように、一気に計画（WBS 構築）することが難しい場合、アジャイル的に WBS を構築していく方法を採るのが良い（具体的には、WBS 構築は開発よりも少し早めにする程度に留める）。さらには、その活動のサイクルに合わせたメトリクス収集・改善を実施するのが良い。

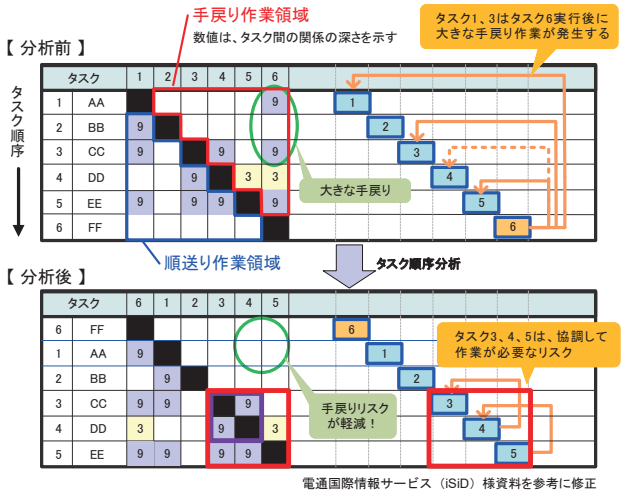


図 24　アジャイルな変更管理を可能とする仕組み

10. ソフトウエア会社がシステムズエンジニアリングプロセスを作れるのか

システムズエンジニアリングの骨格は、アナリシス（機能展開）と、シンセシス（構造統合）である。

機能展開は、抽象化を進めていく作業であり、抽象化を主体に進めていくソフトウェア開発の知識が活かせる。シンセシスは、物理的な構造ノウハウ集（存在するとして）から機能に適合する設計解を探し出し、組合せ最適を探す作業であり、その探し方（プロセス）を決める作業は、抽象的な能力が必要とされる。

このように具象と抽象を行き来する作業プロセスを決めるのは、ソフトウェアでは、クラス設計時に通常行っていることであり、知識分野が違ったとしても、統一プロセスを作るのに必要な能力は変わらない。

11. むすび

V2 Professor は、モデルレベルでのトレーサビリティ整合が取れる標準設計ガイドラインとすることを目標として、10 年にわたる現場での実践から得られた知見を反映してきている。そして、現在も、現場のノウハウ、理論を反映させながら成長し続けている。

設計作業自体、一般的にはパターン化が難しいことから、それを設計ガイドとすることは難しい。事実、説明を多くすれば良いというものではなく、多過ぎるとそれに頼ってしまい考えない人が増える。少な過ぎると、使い物にならないものができあがる。これは、一つには、万人に使えるガイドを作成しようとするからであり、ガイド利用者のスキルレベルを考慮し、スキルの高い人は、ツールで表現方法を統一するための作図ガイドで十分事足りるし、設計知識不足の人には、「設計方法」を教育するための「設計ガイド」が必要であろう。そのようなガイド分割が必要であることも、実践を通じて学んだ。

理想を目指す活動を継続している限りは、近づくはずと考えながら理想の標準を今後も目指していく。

本内容を基に、弊社の設計教育を受講いただき、実践的な題材を提供いただきました様々な分野の社外の方々からのフィードバックが、現場で使える設計モデル作成ルールとなっています。変化する設計ルールを根気強く適用し続け、改善へのヒント、気づきを頂いた多くの方々に感謝の意を表します。

参考文献

(1)　Paul, G., Beitz, W., Feldhusen, J., Grote, K. H.：エンジニアリングデザイン 工学設計の体系的アプローチ，第 3 版，森北出版（2015）

(2) 宇宙航空研究開発機構：JAXA 制御系設計標準，JERG–2–500A（2013）
(3) 宇宙航空研究開発機構：JAXA システム設計標準，JERG–2–100（2016）
(4) 田浦 俊春：創造デザイン工学，東京大学出版会（2014）
(5) 藤原 啓一：要求から詳細設計までをシームレスに行うアジャイル開発手法，MSS 技報，24（2014）
https://www.mss.co.jp/technology/report/pdf/24-04.pdf
(6) 藤原 啓一：大規模組込システムの要求分析、システム方式設計、そして、ソフトウェア設計までをつなぐモデルベース設計手法，MSS 技報，27（2017）
https://www.mss.co.jp/technology/report/pdf/27_03.pdf
(7) 畑 剛，ほか：航空・宇宙における制御，コロナ社（1999）
(8) 廣田 幸嗣，ほか：電気自動車の制御システム，東京電機大学出版局（2009）

執筆者紹介

藤原 啓一
1985 年入社。防衛のソフトウェア開発に従事以降カーナビ、気象レーダ、ニューラルネット、医用画像、衛星通信、業務系システム、通信システム、車載制御システム等のシステム設計、ソフトウェア開発に従事。2015 年から副事業部長。