モデルベースシステムズエンジニアリング（MBSE）のすゝめ
―「はかる」，「つなぐ」を考える―

Model-Based Systems Engineering (MBSE)
- “To Measure” and “To Connect” -

西村 秀和
Hidekazu NISHIMURA
慶應義塾大学大学院　システムデザイン・マネジメント研究科
研究科委員長・教授　工学博士
Professor
Dean of Graduate School of System Design and Management Keio University
Ph. D.

近年の製品やサービスは非常に複雑なものとなっており，企業は限られたコストの中で，環境規制に対処し，安全を確保しなければならず，地域向けにカスタマイズすることが求められることが多い。こうした複雑な製品やサービスなどのシステムを成功裏に実現するために，モデルベースシステムズエンジニアリング（MBSE）のアプローチが注目されている。自動運転システムを含む自動車のような複雑なシステムのライフサイクル全体にわたる確実なエンジニアリング活動全体を実現するには，何をすれば良いか？コンセプト，アーキテクチャおよび設計の定義を行う際には，必ず，検証と妥当性確認が必要となる。「はかる」，「つなぐ」をMBSEの観点から論じる。

In recent years, products and services have become extremely complex, and enterprises must address its environmental regulations, ensure its safety within limited costs and often are required to customize those for the area. Model-based systems engineering (MBSE) approach has been attracting attention in order to realize systems such as complex products and services successfully. What should we do to realize rigorous engineering activities through the entire life cycle of complex systems like vehicles including automated driving systems? When defining concept, architecture and design of a system, verification and validation must be required. “To measure” and “to connect” are discussed from the viewpoint of MBSE.

はじめに

近年の製品やサービスはメカ，電気，ソフトウェアなどから成り，特に自動車は，限られたコストの中で，環境規制に対処し，安全を確保しなければならず，また，地域の市場に合わせてカスタマイズするいわゆる派生開発が求められることが多い。さらにMaaS（Mobility as a Service）[1]をはじめ自動車関連のサービスを積極的に提供する動きが活発になっている。企業はこうした複雑な製品やサービスを成功裏に実現する必要があり，そこに関連するエンジニアリング活動は複雑さを増している。

複雑な製品やサービスなどのシステムを成功裏に実現するために，モデルに基づくシステムズエンジニアリング（MBSE）のアプローチが注目されている[2, 3]。従来の文書を中心としたシステムズエンジニアリングでは，複雑なシステムの記述を十分に行うことは難しく，また，エンジニア間でこれらの関連情報をやりとりすることは容易ではなかった。システムモデルを用いて対象のシステムを記述することで対象のシステムを規定し，システム要求の定義，システムアーキテクチャの定義，設計定義のプロセスを双方向にトレーサビリティがとれる形で進めることができる。また，システムモデル記述は検証および妥当性確認のプロセスで活用できる。

今後，自動運転システムを含めた進化が求められる自動車には，そのライフサイクル全体にわたる確実なエンジニアリング活動の実現が求められている。環境と安全確保の問題は，ますます高いレベルでの解決が求められるものと考えられる。では，そのためには何を準備しておく必要があるだろうか？複雑なシステムの開発を着実に行うためには，コンセプト，アーキテクチャそして設計が，相互にトレーサビリティがとれた形で定義されることが求められる。それぞれのプロセスでは検証と妥当性確認が必要となり，システムをコンポーネントから統合する過程でも同様に検証と妥当性確認が必要となる。さらに近年では，特に

ソフトウェア分野で，製品の運用（Operation）中にそこに組み込まれたソフトウェアを開発（Development）すること（DevOpsという）が求められている[4]。

こうした背景から，システムの検証と妥当性確認を実現するシステムの存在が重要になる。自動車を設計し製造するエンジニアが，自動車あるいは自動車を構成する要素の検証および妥当性確認を行いたいと考えたときに，これを実現するシステムをどのようにつくると良いのか？そこでは，何のために何を「はかる」必要があるのか？何のために何を「つなぐ」必要があるのか？検証と妥当性確認でのテストケースをどのように定めたら良いのか？これらの多くの課題に速やかに解決策を導き出すことは容易なことではないし，これに対する解をMBSEが持つ訳ではない。しかし，MBSEの中でアーキテクチャ定義を行うことは極めて重要な意義をもち得ると考えている。ここではその観点から論じてみたい。

システムズエンジニアリングの開発プロセス[2, 3]

製品やサービスなどは，『1つ以上の定められた目的を達成するために編成された相互作用する要素の組み合わせ』である，いわゆるシステムとして考えることができる。システムは，コンセプトを検討する段階から開発，製造，運用，保守，廃棄にいたるライフサイクルステージを持ち，それぞれのステージで対象のシステムを実現するために必要となるシステムが存在する。すなわち，製品やサービスなどのシステムを実現するには，機械，電気などのハードウェア，これらを適切に動作させるソフトウェアのみならず，人，設備などを必要とし，これらをどのようにマネジメントしていくかが，製品やサービスの成否に大きな影響を与える。

こうした複雑なシステムを成立させるために，主として技術的な面から取り組むためのアプローチあるいは手段として，システムズエンジニアリングがある。近年では，システムが複雑になってきたことから，文書を中心としてシステムの記述を行うアプローチに代わり，モデルを用いたアプローチ，すなわちMBSEが注目を浴びている。ここでモデルには，対象のシステムを記述する範囲を決めるための幅，深さおよび忠実度がある。モデルの幅は，機能，インタフェース，性能，制約，品質特性などのシステム要求の適用範囲を反映する。モデルの深さは，システムコンテキストからシステム要素までの分解方向の範囲である。モデルの忠実度はモデルが表現する詳細さのレベルを示す。このモデルの種類として，形式的モデルには論理モデル，定量モデル，幾何学モデルがある。また，非形式的モデルとして描画および文書があるが，これらには正確さに欠ける曖昧な表現が含まれる可能性があることを考慮する必要がある。

例えば，自動車に搭載されている内燃エンジンは，様々な環境規制が課せられ，極めて複雑なシステムとなっている．このエンジンシステムを実現する過程で，システムズエンジニアリングの初期の段階ではエンジンシステムのアーキテクチャを定義することが重要となる。さらにエンジンテストベンチ上での試験が繰り返し実施され，燃費性能や排ガス性能の適合，テストが行われる。このため，エンジンテストベンチはエンジンを実現するために必要なシステムということができる。そして，このエンジンテストベンチを開発する際には，エンジンシステムの初期の段階で定義したアーキテクチャを参照することができる。このようなあるシステムを実現するために必要なシステムを，有効にするシステム（enabling system）と言う。対象とするシステムをテスト，あるいは分析するために必要となるMIL（mode-in-the-loop），SIL（simulation-in-the-loop），およびHIL（hardware-in-the-loop）シミュレーションまたはテストには，これを実現するための有効にするシステムが必要となる。

Figure 1にエンジンシステムの開発プロセスの全体を表す2元V字モデルを示す。**Figure 1**は，開発対象とする製品の分解と統合を表す垂直方向の「アーキテクチャV」と，システム，サブシステム，コンポーネントのそれぞれの開発プロセスである，要求分析，アーキテクチャ定義，設計仕様の決定，製作，検証（Verification），妥当性確認（Validation）を表す水平方向の「エンティティV」（**Figure 2**）とを同時に表す2元V字モデルである[5]．

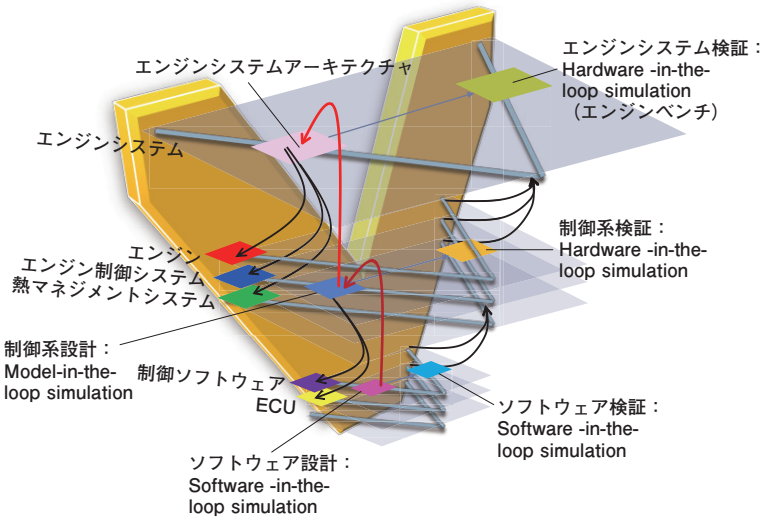

Figure 1　Dual Vee model of development process for an engine system

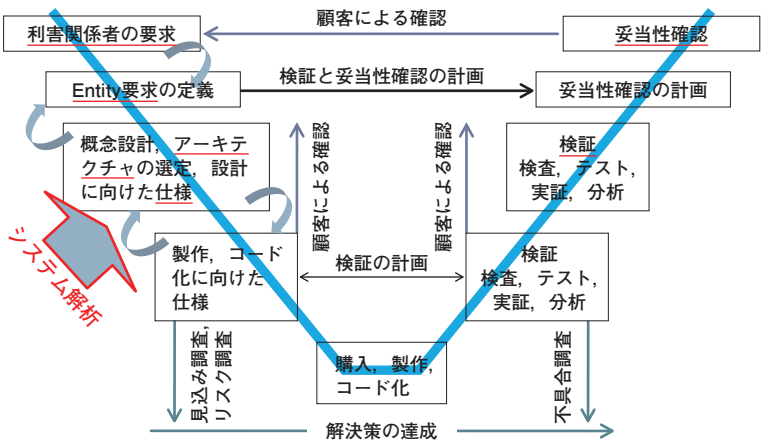

Figure 2　Entity Vee in horizontal direction of Dual Vee

Figure 2のエンティティＶでは，利害関係者要求から始まり，システム要求の定義，アーキテクチャの定義，設計定義へと詳細化が進む。その際，繰り返しこれらの定義の関係性が正しいことを検証して進めることが重要であり，これらのアクティビティの結果としてシステムを構成する実装可能なサブシステムを定義することができる。そして，要求に従ってつくられて検証されたサブシステムを統合しシステムとしての検証および妥当性確認を行う．サブシステムとコンポーネント間の関係性も同様となる．なお，検証および妥当性確認の計画は，あらかじめFigure 2の左側のプロセスで行っておく必要がある。

アーキテクチャに基づく有効にするシステムの定義

対象とするシステムのアーキテクチャを定義するには，抽出された利害関係者の関心事を捉えたアーキテクチャビューポイントを持ち，それが決定するアーキテクチャビューを持ったアーキテクチャ記述が必要となる。システムアーキテクチャは，ある環境中で，システム要素とその関係性から具体化されたシステムの基本概念または特性であり，システムを設計し，進化させる際に原則となるものである[6]。したがって，定義されたアーキテクチャは，その後の設計以降のプロセスに対して制約を課すこととなる。また，アーキテクチャビューに応じてモデルを用いてアーキテクチャを記述することになる。アーキテクチャはシステムが用いられる環境についても限定していることに注意されたい。

事例として，エンジンシステムのアーキテクチャをコンテキストレベルで記述したアクティビティ図をFigure 3に示す。ここでは，SysML(Systems Modeling Language)[7]を用いて記述している。このダイアグラムはある環境中でのエンジンシステムのコンテキストを表す。ドライバの指令に応じてエンジンシステムが生成したトルクが，トランスミッションを介してドライブトレインに伝わり，さらに道路へトルクを伝えていることを示している。自動車が走行するために必要な仕組みの基本機能を記述するダイアグラムであるが，空気を提供する大気とタイヤと接する路面も定義しており，これらは自動車が走行する上で必要な環境となる。

Figure 3をもとにエンジンシステムをテストするための有効にするシステムを考えた上で記述した結果をFigure 4に示す。すでにこれはエンジンテストベンチとして知られるものであるが，Figure 3に示すダイアグラムを含めたエンジンのシステムモデルを記述することから，エンジンシス

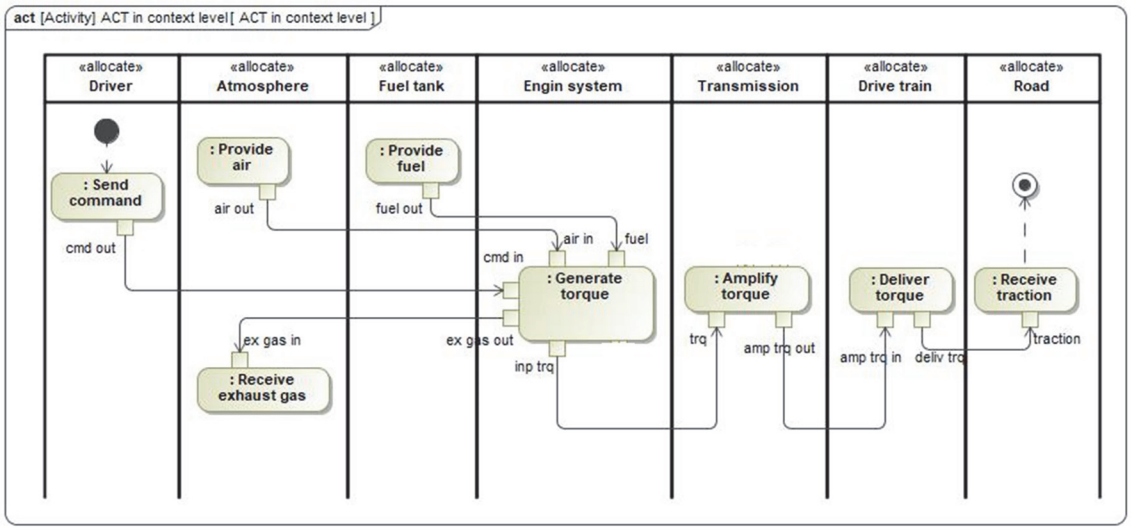

Figure 3　Activity diagram describing the context-level behavior of an engine system

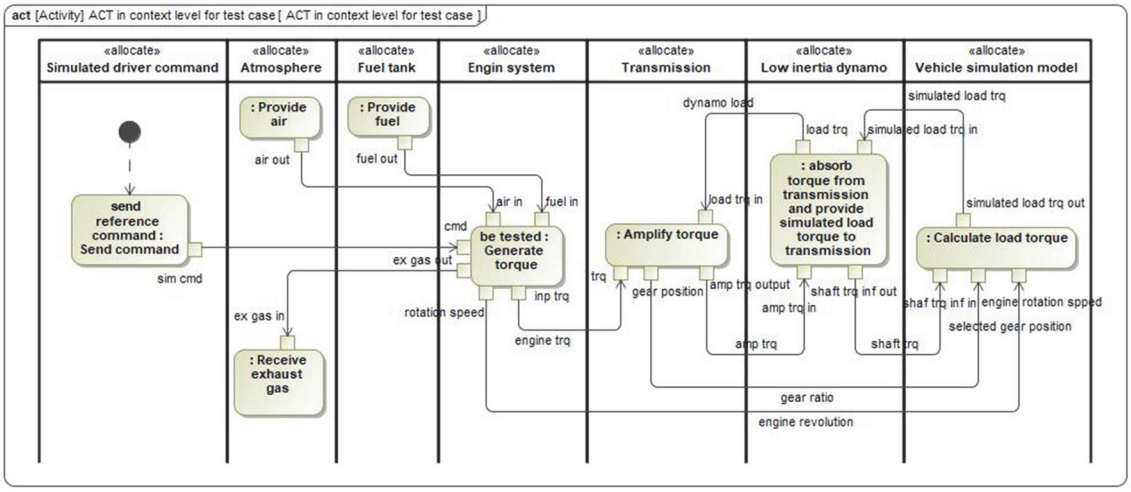

Figure 4　Activity diagram describing the behavior of the engine system and its test system

テムをテストするために，どのような状況で何をはかる必要があるかを規定することで，エンジンテストのためのシステムを規定できる。**Figure 4**にはエンジンテストを実施するためのシステム構成として，模擬したドライバ指令（Simulated driver command），車両シミュレーションモデル（Vehicle simulation model），低慣性ダイナモ（Low inertia dynamo）が示されている。また，燃料タンク（Fuel tank）および大気（Atmosphere）では環境性能の評価に「はかる」が関与する。

「はかる」と「つなぐ」：検証と妥当性確認

エンジンシステムの検証と妥当性確認

エンジンシステムの仕様に対して検証を行う際には，車両がどのような環境に置かれ，どのような走行をドライバからの指令で実施する必要があるのか，いわゆるシステムコンテキストの理解の上での仕様の検証を行う必要がある。例えば，エンジンへ供給する空気の特性が関与する可能性があり，またどのようなドライバの指令を模擬する必要があるのかが，関係する。さらに車両のシミュレーションモデルがどこまで精緻に，忠実度の高いモデルになっているのか，あるいはそうではないのかも十分に配慮される必要がある。特に，路面状況が異なる条件で検証が必要な場合には，車両が有するタイヤモデルの忠実度が大きな影響を与える可能性がある。

妥当性確認をとる必要がある場合には，開発対象のエンジンが何に対して妥当であることが求められているのかを明確にしておく必要がある。たとえば，ユーザーが「満タンでできるだけ長く走行できること」をニーズとして持っている場合に，そのユーザーニーズに合致したユーザー要求と運用シナリオを定義し，その運用シナリオ上でユーザー要求に対する妥当性を確認することとなる。システムの妥当性確認のプロセスでは，製品が利害関係者要求，ミッションの輪郭，運用シナリオを充足していることを実証することが求められる[2, 3]。

ユーザーニーズに合致したユーザー要求「ユーザーは自動車を満タンにしたあと，無給で少なくとも500 km離れた場所まで運転していけること」と，その運用シナリオを定義したものすると，走行経路に基づき大気と路面を定義し，模擬したドライバ指令を用意することが求められる。車両シミュレーションモデルの忠実度を高くする場合には，燃料タンク内の燃料の量に依存して車両重量を可変とする必要があるだろう。また，ドライバ指令についても，ユーザーの運転技量を反映するなど忠実度を増すことが求められることもある。これらの決定に際しては，MBSEのアプローチにより，定義されたシステム要求，アーキテクチャ，および設計に基づくこととなる。

自動運転システムのテスト

自動車事故およびそれに起因する死傷者を減らすこと，交通流を良くすることによる渋滞の緩和など，自動車に搭載される自動運転システム（Automated Driving System：ADS）に対する期待は大きい。SAE（Society of Automotive Engineers）で定義されたレベル3の自動運転では，自車両の制御や交通環境の監視といった動的運転タスク（Dynamic Driving Task）をADSが実施する．また，予め定められている運用設計領域（Operational Design Domain）を自動運転車が逸脱した場合，運転権限がADSからドライバに移譲され，ドライバは適切に対処しなければならないとされる[8]．

ISO/PAS 21148[9]は，Safety of the intended functionality（SOTIF）に関する標準であり，わかっている安全（known safe）／わかっていない安全（unknown safe）／わかっている不安全（known unsafe）／わかっていない不安全（unknown unsafe）の4つのシナリオのカテゴリーを

定義し，この中の「わかっていない不安全」シナリオの領域をできるだけ減らすことの重要性を述べている。わかっていないが安全なシナリオであれば，そのままにしておいて良いが，不安全になってしまう，わかっていないシナリオが存在することは，極めて危険である。この「わかっていない不安全」シナリオの領域を減らすため，残余リスクを評価するための複数の方法で妥当性確認を行うこととしている。

運用シナリオを作る際にはそのシナリオに登場する人や物に漏れがあってはいけない。NHTSA（National Highway Traffic Safety Administration）では，これを定義するための枠組みが発行されており，そこでは自動運転にかかわる外部システムの分類が示されている[10]。仕様書で規定された範囲（パラメータの限定を含む）での検証を十分に実施したことを前提として，運用シナリオをもとに行う妥当性確認では，利害関係者ニーズから考えることにより，生じてしまう可能性のある事象を発生させる必要がある。

たとえば，利害関係者ニーズとして，「早く目的地に到着したい」があった場合に，実際にシステムを利用するユーザーは，このニーズをもとにシステムを利用することが考えられる。その結果，交差点の左折時に，ユーザーが公道ではない道路を通過することを求める場合がある。このようなユーザーのニーズに対して対応することを許容するのか否かを，担当エンジニアは設計時に考慮していない可能性がある。システムに対する適切な妥当性確認ができる運用シナリオを用意しておくことで，このケースでの残余リスクを評価できることとなる。このためには，テストシステムに，シナリオに登場する様々な人，物を「つなぐ」ことが求められる。

まとめ

製品やサービスを有効にするテストシステムに必要となる「はかる」，「つなぐ」を，モデルベースシステムズエンジニアリングの観点から論じた。開発対象のシステムを検証し妥当性確認をとる際に，どのような環境で何を「はかる」ことが要求されているのかを定義しておくことが重要であること，運用シナリオに基づき妥当性確認を行うには，利害関係者ニーズを反映した運用シナリオを定義し，テストシステムの中で様々な人，物を「つなぐ」必要が生じる。これらの一連の検証と妥当性確認を実現するには，システム設計定義のもととなるアーキテクチャの定義がキーポイントとなる。開発対象であるシステムのアーキテクチャをモデルで記述することで，比較的容易にテストシステムの構築に結びつけることができる。

今後は，開発のステージのみで検証と妥当性確認が行われるのにとどまらないと考えられる。開発ステージでは繋がっていなかったものが運用中に繋がってきたときに，運用中のシステムを改修するなどして，テストをする必要性が生じる可能性がある。すでにソフトウェア分野では，DevOpsが行われていると言われる。そのテストをどのように実施するのか？「はかる」，「つなぐ」は今後，必要不可欠なものとなる。

参考文献

[1] Jana Sochor, Hans Arby, I. C. MariAnne Karlsson, Steven Sarasini, A topological approach to Mobility as a Service: A proposed tool for understanding requirements and effects, and for aiding the integration of societal goals, Research in Transportation Business & Management, 27, 3-14, 2018

[2] INCOSE Systems Engineering Handbook 4th Edition, WILEY, 2015

[3] システムズエンジニアリングハンドブック第4版，監訳：西村秀和，慶應義塾大学出版会，2019（[2]の翻訳書）

[4] Michael E. Porter, James E. Heppelmann, How Smart, Connected Products are Transforming Companies, Harvard Business Review, Oct. 2015

[5] Kevin Forsberg, Hal Mooz, Howard Cotterman, Visualizing Project Management, Third Edition, John Wiley & Sons, Inc.

[6] INTERNATIONAL STANDARD ISO/IEC/ IEEE 42010, First edition, 2011-12-01

[7] システムズモデリング言語 SysML（A Practical Guide to SysMLの翻訳書），監訳：西村秀和，東京電機大学出版局，2012

[8] SAE International：Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_201806（2018）

[9] PUBLICLY AVAILABLE SPECIFICATION ISO/PAS 21448, First edition, 2019-01

[10] A Framework for Automated Driving System Testable Case and Scenario, NHTSA, Sept. 2018