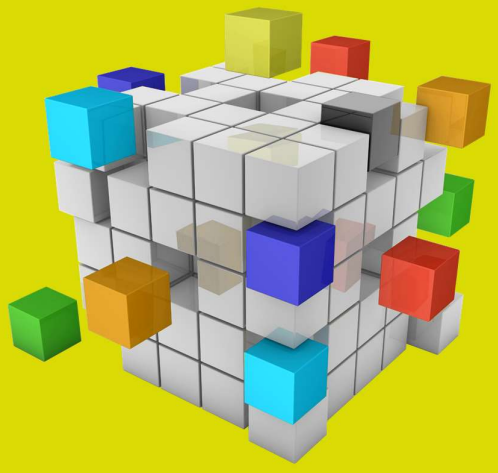
invenio



Whitepaper

アーキテクチャ・セントリック、モデルベースドシステムズエンジニアリング ARCHITECTURE-CENTRIC, MODEL-BASED SYSTEMS ENGINEERING (ACMBSE)

バージョン	日付	著者	変更点
1.2	21.08.2024	Enrico Seidel	V.1.2ドイツ語からの翻訳

invenio

内容

サマリー

過去10年間で、自動車産業は従来のドキュメントベースのシステムズエンジニアリングから、よりアーキテクチャ・セントリックでモデルベースドシステムズエンジニアリング（ACMBSE）への大きな変革を遂げました。この変革は、急速に増大する複雑さと、従来のドキュメントベースの方法ではシステム記述を管理することが困難になったことによって促進されました。このホワイトペーパーでは、著者が開発したアーキテクチャフレームワークを使用して、MBSEがどのようにこの変革を支援できるかを探ります。著者は、複数の視点に基づいたアーキテクチャ・セントリックなシステムズエンジニアリングアプローチを提案しています。アーキテクチャの中心性に焦点を当て、モデルを使用することで、複雑なシステムの分析、開発、設計がより効率的かつ効果的に行えるようになります。また、著者は自動車業界からの事例を用いて、このアプローチがどのように開発ワークフローに実践的に統合できるかを示しています。

1 はじめに

他の産業分野と同様に、自動車産業は今日の急速に変化する環境において、生産性、収益性、人的資本に関する課題に直面しています (Nayak et al., 2022)。ソフトウェアベースのエコシステムへの移行により、自動車メーカーは複数の機能領域において大幅な技術的進展を達成する必要が生じました (Burkacky et al., 2021)。システム機能の強化と開発効率の向上に対する需要の高まりが、システムモデルの使用を促進しています。従来のドキュメントベースのシステムズエンジニアリングでは、回復力があり安全性が重要なシステムにおけるソフトウェア駆動の機能の複雑さを管理するには不十分です。
その代わりに、ISO 26262（ISO 1 2018）などの重要な基準への準拠を促進するために、MBSEアプローチが必要となります。

本論文の著者はさらに一歩進み、アーキテクチャを中心に据えたアプローチを提唱しています。従来のドキュメントベースのシステムズエンジニアリングは、自動車システムやその部品のような現代の複雑なシステムに対して不十分であることが証明されています。そのため、本論文で述べるアーキテクチャ・セントリック（中心）モデルベースドシステムズエンジニアリング（ACMBSE）が重要性を増しています (Brooks et al., 2007)。ACMBSEは、厳密なモデルとシミュレーションを通じてシステム動作の設計と実装を保証する手段を提供し、自動車業界のサプライヤーなどの組織が変化する需要に適応するための重要な方法論となります。

本稿では、ACMBSEの採用について検討し、このアプローチがシステム分析、機能開発、論理要素の設計、および技術システム設計にどのように使用できるかを示します。さらに、電子制御ユニットの実装やその物理的な実現にまで拡張されることも論じます。

2 アーキテクチャ・セントリック MBSE

アーキテクチャ・セントリックモデルベースドシステムズエンジニアリング（ACMBSE）は、複雑なシステムの開発において、システムアーキテクチャの中心的な役割を強調するシステムズエンジニアリングアプローチです。図1に示されているように、ACMBSEは、システムとそのサブシステムのアーキテクチャが、コンポーネントやそれらの相互関係、ソフトウェアおよびハードウェアの両面を含むシステム全体の動作を表す統一的な概要を提供するという原則に基づいています。したがって、ACMBSEは、異種でドメイン固有の工学分野への架け橋として機能します。このアプローチは、システムズエンジニアリングの他のすべての側面、例えば要求事項、機能安全、セキュリティおよびプライバシー、システムの可変性、信頼性、検証および妥当性確認が、アーキテクチャと整合し、その視点に寄与することを要求します。

ドキュメントベースの開発と比較して、ACMBSEは以下のような複数の利点を提供します。コミュニケーションの改善、情報の維持管理の向上、システムの複雑さを管理する能力の強化、製品品質の向上、そして優れた知識保持能力です。

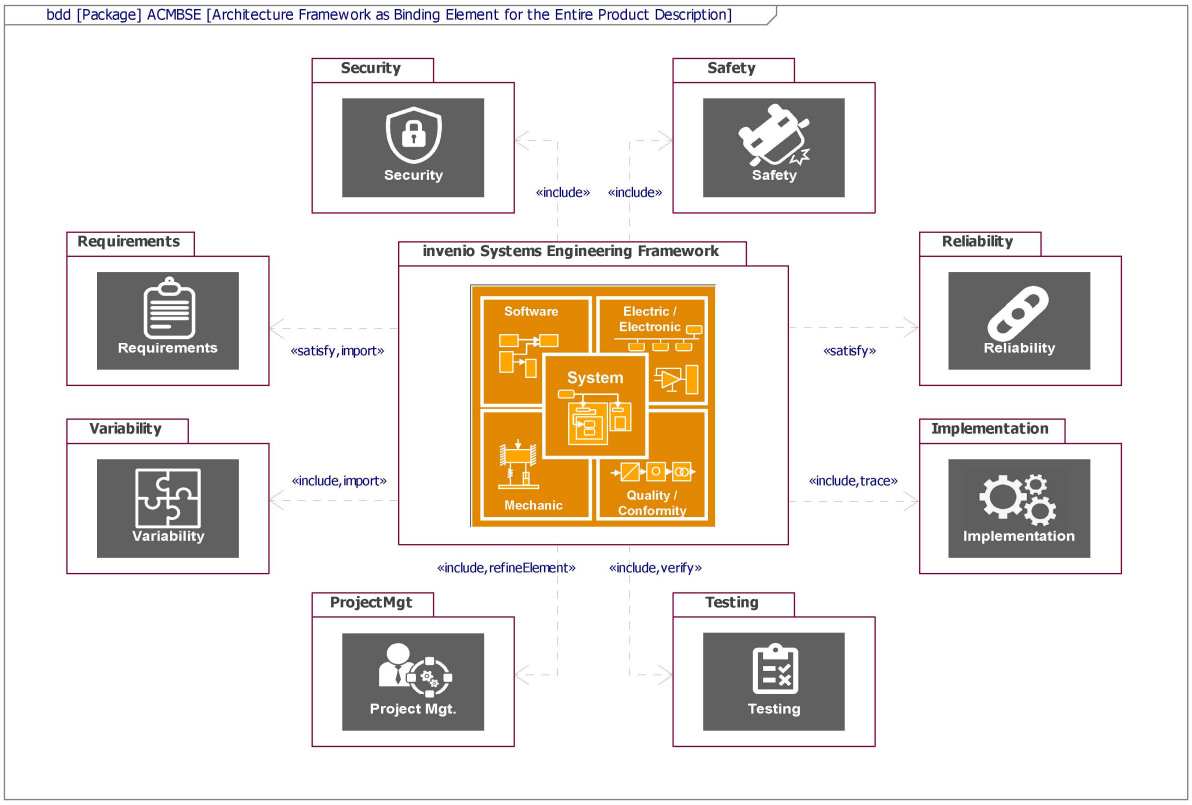


図1: 製品全体の記述をつなぐアーキテクチャフレームワーク

3 アーキテクチャフレームワークの重要性

SysML（OMG 2018）は、モデルベースの開発を可能にするシステムズエンジニアリングで広く使用されている言語です。しかし、特定のドメイン内でシステムアーキテクチャや設計を構築するためのルールや手法を定義していません（ISO 2022）。

クロスドメインプラットフォームを実現するためには、システム製品とそのアーキテクチャ成果物を開発するための標準化された方法とツールのセットが不可欠です。

Harmony（Douglas 2017）、OOSEM（Friedenthal et al., 2015）、MOFLT（Ducamp et al., 2022）のような構造化されたフレームワークの使用は、大規模なMBSEの一貫した実施に不可欠です。各フレームワークは、UPDM（UPDM 2017）などのドメインに特化したアーキテクチャフレームワーク内で強みを持っています。UPDMは、DoDAF（Department of Defense Architecture Framework）とMoDAF（Ministry of Defense Architecture Framework）を組み合わせたものです。しかし、著者はこれらのフレームワークが主に防衛産業向けに開発されたため、自動車産業の特定の要求を十分に満たしていないことに気付きました。再利用性、自動化されたモデル検証、モデル上の視点の体系的な使用、システム抽象レベル、ACMBSEアプローチの前提条件としての階層深度などの課題は十分に対応されていませんでした。エンタープライズの視点を捉えることを目的とするTOGAF（TOGAF 10）やZachman（Zachman 2023）といったフレームワークも、自動車産業の技術的側面に適合させるためには調整が必要です。このような状況下で、唯一SYSMOD（Weilkiens 2020）フレームワークが著者の要求を満たし、カスタマイズされた視点とモデリング技法を提供しました。

著者は主に自動車業界向けに自身のフレームワークを使用することを意図し、電子制御ユニットおよびコンポーネントの開発を含む包括的なシステム設計のために、上記の基準を含める必要があることを認識していました。

これらの理由から、著者は以前、コンチネンタルAGでアーキテクチャフレームワーク（CAF）を開発しており、invenio AGへの移行に伴い、これがさらに大幅に発展し、現在はinvenioシステムズエンジニアリングフレームワーク（iSEF）として公開される予定です。このフレームワークは最近、システム、ソフトウェア、およびハードウェアエンジニアのためにアーキテクチャと設計仕様をシームレスに開発するために必要なすべての方法とプロセスステップを定義する、成熟したモデリング手法として進化しました。SysML言語を拡張する通常のルールや手法に加え、iSEFは、自動車機能やプラットフォームの標準化された開発を支援し、市場投入までの期間を短縮するために、事前定義されたアーキテクチャ成果物、事前開発された設計要素、および再利用可能な機能の包括的なセットを提供することを目的としています。さらに、iSEFはISO 42010（ISO 2022）やISO 15288（ISO 2015）などのすべての重要なISO基準に完全に準拠しています。

iSEFの実装には、モデリングツールとしてIBM Rhapsody（IBM 2022）が選ばれました。これはIBM Engineering Lifecycle Management（ELM）プラットフォームに組み込まれ、SysMLに基づいた完全にカスタマイズされたドメイン固有のプロファイルを使用しています。

4 システムアーキテクチャの3つの次元

システムがますます複雑化する中（INCOSE　2022）、単一のシステムエンジニアがその機能、特性、およびパフォーマンスを包括的に理解することは困難になっています。そのため、システムはより小さな単位やサブシステムに分割して整理する必要があります。各サブシステムには特定のタスクがあり、アーキテクトや設計者はその部分の詳細に集中することが可能です。このような複雑なシステムの例として、車両や航空機があり、多くの組織単位がシステムの異なる部分に取り組み、全体のシステムアーキテクチャを構築しています。

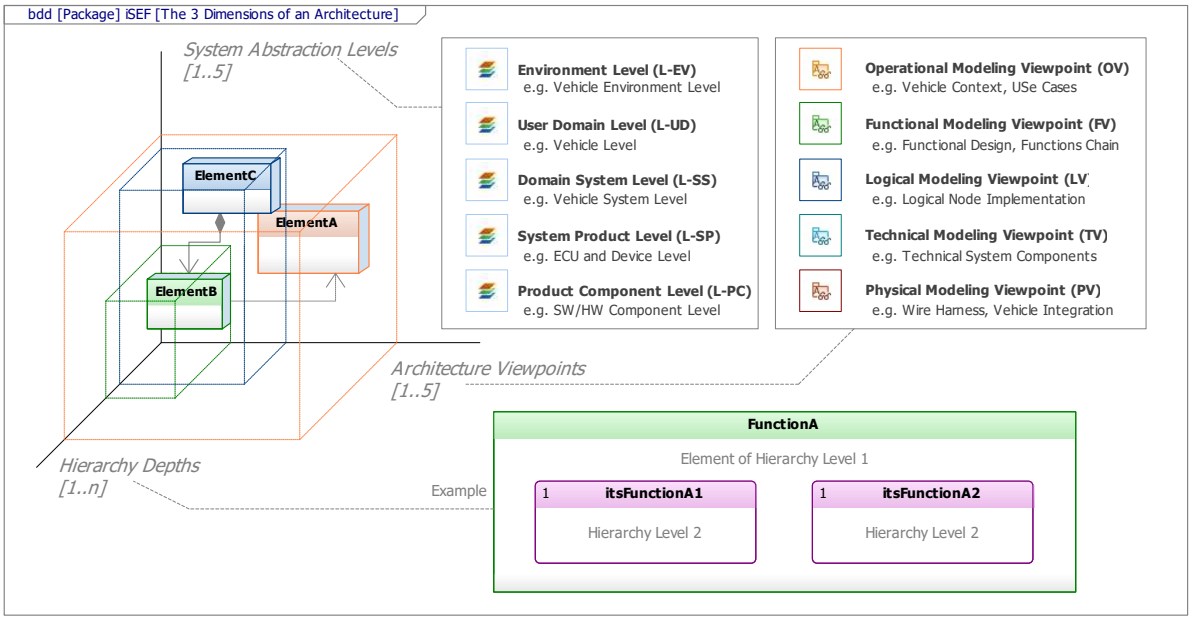


図2: システムアーキテクチャ記述の三つの次元

著者はiSEFを用いて、システムアーキテクチャの3つの次元を紹介しています。それは、システムに対するモデルの視点、システムの抽象レベル、そして階層の深さです。図2に示されているように、これらの3つの直交する次元は、自動車分野に適用され、一貫性のあるシステムアーキテクチャ開発のための構造化されたアプローチを可能にします。

5 iSEFのモデル視点

開発組織のニーズに応えるためのフレームワークを開発する一環として、著者は、再利用性に重点を置いてシステムを分析、定義、作成するために、5つの重要なアーキテクチャ関連モデル視点（図3に示す）を特定しました。これらは、さまざまなステークホルダーの関心に対応することを強調しています。モデルのオペレーショナル・ビューポイント（運用視点）、ファンクショナル・ビューポイント（機能視点）、ロジカル・ビューポイント（論理視点）、テクニカル・ビューポイント（技術視点）、フィジカル・ビューポイント（物理視点）が導入されています。直接アーキテクチャに関連する視点は以下のように定義されています。

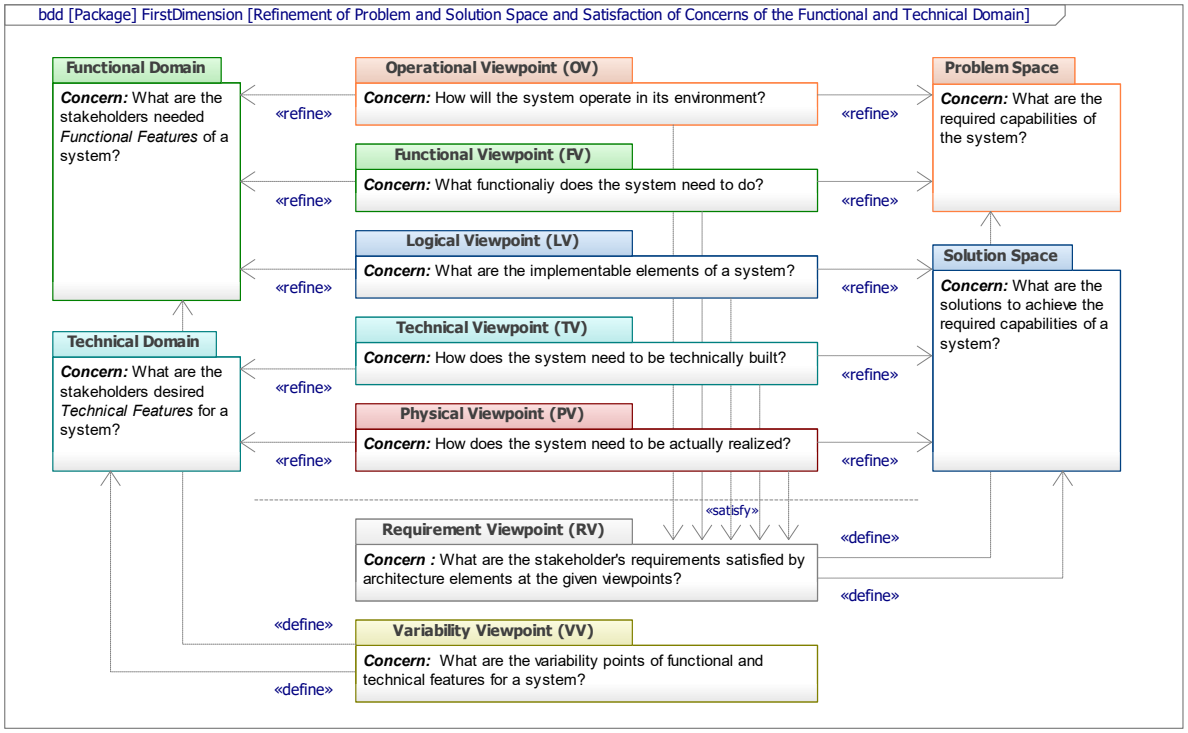


図3: ビューポイントと問題空間/ソリューション空間の関係

オペレーショナル・ビューポイント（OV）– ″運用視点″:

問題空間の一部として、オペレーショナル・ビューポイント（OV）は、運用コンセプトに基づいて（ISO 2 2018）、開発されるシステムを分析することに焦点を当てています。OVは、システム・オブ・インタレスト（対象システム: System of Interest, SoI）をその文脈環境内で定義し、サブシステムとインターフェースをポートを通じて特定し、通信の論理的および技術的側面を表現します。OVの主な目的は、システムのプロセスとユースケースを視覚的に表現し、システムの意図された操作と動作を詳細化し、システム活動にまで落とし込むことです。さらに、OVはシステム要素間の相互作用の順序を概説し、システム・オブ・インタレスト内での通信がどのように行われるかを明確にします。

ファンクショナル・ビューポイント（FV）– ″機能視点″:

ファンクショナル・ビューポイント（FV）は、問題空間の一部として、システム分析に基づいた機能設計に焦点を当てています。FVは、活動や状態遷移図に基づく動作モデルをカプセル化した統合された再利用可能な機能ブロックを定義します（これはOVにおける運用分析の結果です）。FVは、機能間の情報の流れを容易にするために、構造化された標準化された機能インターフェースとその公開された操作を定義するのに役立ちます。さらに、FVは、事前に定義された分解戦略に従って機能ブロックを可視化し、システム・オブ・インタレスト（対象システム，SoI）の全体的な機能動作を包括的に理解することを可能にします。

ロジカル・ビューポイント（LV）– ″論理視点″:

ロジカル・ビューポイント（LV）は、ソリューション空間への移行を示し、与えられた制約やプラットフォームを考慮しながら、対象システムの実装に焦点を当てます。LVは、ファンクショナル・ビューポイント（FV）で定義された機能ブロックを実装可能な単位であるロジカルノードにグループ化します。主な目的は、ノード間の接続や標準化された信号およびサービスを使用して、ロジカルインターフェースを定義または再利用することで、実装の視点からシステムを視覚的に表現することです。さらに、LVは特定のシステム環境内でのアプリケーションやドライバーなどのロジカルユニットの詳細な設計を示し、与えられた非機能要件を考慮します。

テクニカル・ビューポイント（TV）– ″技術視点″:

ソリューション空間の一部として、テクニカル・ビューポイント（TV）は、技術システムを構築するために必要な制約や技術に焦点を当てます。TVの主な目的は、ロジカルノードとその統合された機能のための実行環境を提供することです。TVは、ロジカル・ビューポイント（LV）からロジカルノードを取得し、それらを必要なハードウェア（HW）およびソフトウェア（SW）インターフェースと接続し、実行可能な機能、ロジカルノード、技術コンポーネントを結びつけます。TVは、HWとSWの要素を橋渡しし、特定のハードウェア固有の境界や技術的制約内でシステム設計を統合します。

フィジカル・ビューポイント（PV）– ″物理視点″:

ソリューション空間におけるフィジカル・ビューポイント（PV）は、利用可能な物理的な製品やコンポーネントを使用して、最終的にエンドユーザー向けのシステムやサブシステムを実現することに焦点を当て、非機能的な性能や品質要件が満たされることを確保します。この視点は、開発されたまたは既存の技術的/物理的コンポーネントや要素を用いたシステム統合に関するハードウェアアーキテクチャのブロック図を視覚化することを可能にします。PVはまた、電気/電子（EE）アーキテクチャとEEの詳細設計（例:　ハードウェア回路図）との密接な連携を確立し、E/E開発プロセスを最適化します。

要求のモデリング（リクワイアメンツ・ビューポイント，Requirements Viewpoint）および変動性モデルの作成（バリアビリティ・ビューポイント，Variability Viewpoint）は、ISO 26550（ISO 1 2015）およびISO 26580（ISO 2021）で定義されており、図3に示されるように、モデルの視点に2つの追加のビューポイントを拡張し、これらも開発プロセスの不可欠な部分となっています。さらに、間接的にアーキテクチャに関連する視点は以下のように定義されています。

リクワイアメンツ・ビューポイント（RV）– ″要求視点″:

リクワイアメンツ・ビューポイント（RV）は、ジグザグ手法（Weilkiens 2020）を使用して他のビューポイントと連携し、モデル内でテキスト要求を管理し、表現します。RVは、仕様書での顧客要求や要求仕様書でのシステム要求を定義し、システム分解後のシステム要素要求の定義を管理し、要求とアーキテクチャ要素の間のトレーサビリティをサポートします。RVはiSEFアーキテクチャモデルにおいて重要な役割を果たし、従来のシステムズエンジニアリングとACMBSEをつなぐ橋渡しとなります。特に、モデルがサプライヤーとOEM間で交換できない場合、システムを要求ベースの設計仕様によってのみ調整できる場合に、RVは特に重要となります。

バリアビリティ・ビューポイント（VV）– ″変動性視点″:

バリアビリティ・ビューポイント（VV）は、ステークホルダーやエンドユーザーの視点からシステムの機能的および技術的な特性（フィーチャー）を記述するフィーチャーカタログに基づいて、システムのバリアント（派生型）を管理します。VVモデルはISO 26550（ISO 1 2015）に従い、依存関係が階層的かつ変動性に基づいて整理されています。VVは、アーキテクチャ要素におけるバリエーションポイントを使用して、与えられたフィーチャーに従ってシステムの変動性をフィルタリングします。VV内の変動性モデルは、ユーザーが認識できる機能や技術的ソリューションに対するアーキテクチャ要素の可視性に影響を与えます。VVは、特に多くのバリアントを持つ複雑なシステムにおいて、プラットフォームや製品ラインの管理に不可欠であり、エンジニアリング組織内での再利用性を高めるのに役立ちます（Forlingieri 2022）。

基盤となるiSEFモデリングプロファイル（iSefML）は、SysML言語に基づいており、先に述べたビューポイントで使用されるすべてのモデリングをサポートし、可変なシステム開発プロセスに最適化されています。このプロファイルには、カスタマイズされたモデルブラウザが含まれており、必要な要素の使用に合わせて変更された描画ツールを使用し、モデリングプロセス中にモデル要素間の構成および相互作用の正確性を保証するインタラクティブなルールセットを実装しています。

6 iSEFの抽象化レベル

システムの技術的および機能的な範囲、ならびにシステムがどの程度まで詳細に分析されるべきかは、システム抽象化レベルによって定義されます。これらはシステムの本質的な特性ではなく、人工的な境界を形成しており、任意のシステム（ここでは車両関連システムに焦点を当てています）を分類またはカテゴリ化するものです（図4に示されています）。システム抽象化レベルは、物理的、空間的、または時間的な詳細の抽象化や一般化のプロセスを指す場合もあれば、ユーザーが認識するモデルや現実世界の情報の表現に焦点を当てる抽象化を指す場合もあります。iSEFでは、図4に示されたフレームワークに従って、システムのドメインに適用される5つの抽象化レベルが定義されています。これらは以下の通りです。

エンバイロンメント・レベル （L-EV）–“環境抽象化レベル” :

環境抽/エクスターナル・ビュー・レベル（L-EV）は、システム・オブ・インタレスト（System of Interest, SoI）の文脈を含みます。例えば、車両環境やモバイル通信システムが、直近の運用環境とどのように相互作用するかが該当します。L-EVは、ユーザーとの相互作用や、システムが反応または刺激を与えることができる環境効果に関係しています。さらに、この抽象化レベルには、システム・オブ・システムズ（System of Systems, SoS）内での独立したシステムや機能の相互作用のモデリングも含まれます（例: 交通管理システム、フリート管理システム、クラウドソリューション）。

ユーザ・ドメイン・レベル （L-UD）–“ユーザ領域抽象化レベル”:

ユーザ・ドメイン・レベル（L-UD）は、ドメイン内におけるシステムおよびその構成サブシステム（例: 車両、スマートフォン、飛行機など）を含みます。L-UDは、ユーザーが直接操作するシステムや、環境の影響に直接反応できるシステムに焦点を当てます。このレベルでは、システムの物理的なサイズや複雑さは決定要因ではありません。例えば、飛行機は次の抽象化レベルにおいて、スマートフォンよりも著しく多くのサブシステムを持っていますが、このレベルではシステムの物理的なサイズや複雑性ではなく、システム全体がどのように機能し、ユーザーのニーズに応え、環境とどのように相互作用するかが強調されます。

ドメイン・システム・レベル（L-DS）–“領域系統抽象化レベル”：

ドメイン・サブシステム・レベル（L-DS）は、特定の機能、通信、または技術に焦点を当て、エンドユーザーには直接見えなくなるユーザーシステム内のサブシステム（例:　高度運転支援システム）を含みます。この抽象化レベルでは、統合的なサブシステムは通常、少なくとも2つ以上のシステム製品で構成されます。例としては、車両のドアに含まれる機械、電気機械、および電子部品のすべてや、スマートフォンのモバイル受信システムが挙げられます。L-DS抽象化レベルは、これらのサブシステムが内部でどのように相互作用し、全体のシステムの他の部分とどのように連携するかを理解するために重要であり、システム全体の機能性と信頼性を確保するために重要な役割を果たします。

システム・プロダクト・レベル（L-SP）–“系統製品抽象化レベル”：

システム・プロダクト・レベル（L-SP）は、車両ドメインなどのサブシステム内に存在するすべての技術システムを含みます。このサブシステムは、すべての分野固有のコンポーネント（ソフトウェア、電気/電子、機械）を統合して物理的なシステムとして製品化することによって定義されます。これには、通常、制御ユニット（ECU）、技術的アクチュエータ、センサー、およびこれらのシステムに統合される機能が含まれます。L-SPでは、これらの多様なコンポーネントがどのように集まり、現場に展開できる機能ユニットを形成するかに焦点が当てられています。L-SPで統合されるこれらの機能は、特定の性能、安全性、および信頼性基準を満たすように設計されており、このレベルはサブシステム全体が期待通りに機能することを保証する上で非常に重要です。

プロダクト・コンポーネント・レベル（L-PC）–“製品構成要素抽象化レベル”：

プロダクト・コンポーネント・レベル（L-PC）は、単一の分野内で記述されたすべてのシステムコンポーネントを含み、システム製品を実現します。これには、ハードウェアとソフトウェアのインターフェースや、コンポーネント間の通信プロトコルなどの学際的なインターフェースも含まれます。分野固有のコンポーネントは、その後、特定のハードウェアおよびソフトウェアアーキテクチャの記述に向けたさらなる分解の出発点となります。このレベルでは、各要素の詳細な設計と仕様が考慮され、分野固有のコンポーネントとそのインターフェースが、後続のハードウェアおよびソフトウェアアーキテクチャのさらなる分解と開発の基盤を設定します。

すべてのシステムおよびそのサブシステムは、いずれかのシステム抽象化レベルに分類できます。システムが分解される際、分解の結果として得られたサブシステムは、追加の階層が適用されない限り、元の親システムよりも低いシステム抽象化レベルと見なされる場合があります。開発は常にシステムコンテキストのシステム抽象化レベルから始まりますが、L-EVレベルを必ずしも含める必要はありません。システム・オブ・インタレスト（SoI）は、コンテキストにどのシステム抽象化レベルを使用すべきかを決定し、そのため開発の出発点となります。

システム抽象化レベルの使用にはいくつかの利点があります。各抽象化レベルでは、新しいシステムコンテキストが導き出され、機能、ノード、コンポーネントなどのシステム要素間の明確な相互作用を可能にします。要素間の通信（相互作用）は同じ抽象化レベルでのみ許可されており、異なる抽象化レベル間でのシステムやコンポーネントの「レベル飛び」を防ぎます。

同様に、システム抽象化レベルは、各抽象化レベルで使用できる要素メタクラスを正確に指定するルールを備えた分解モデルを可能にします。これにより、iSEFプロファイルがサポートするツールによって、アーキテクチャ記述内のすべてのアーキテクチャ要素に対して機械的に検証可能なルールを実装することができます。これにより、システムモデリングが有効であり、事前に定義された分解および統合のスキームに従っていることが保証されます。この手法は、異なる分野や企業（例: サプライヤー）のシステムエンジニア間のコミュニケーションを促進します。彼らは関連する抽象化レベルに集中することができ、要素の交換には明確な引き継ぎルールが適用されます。また、各抽象化レベルには対応する仕様および要求事項が必要であるため、対応する仕様および要求事項モデルも必要です。

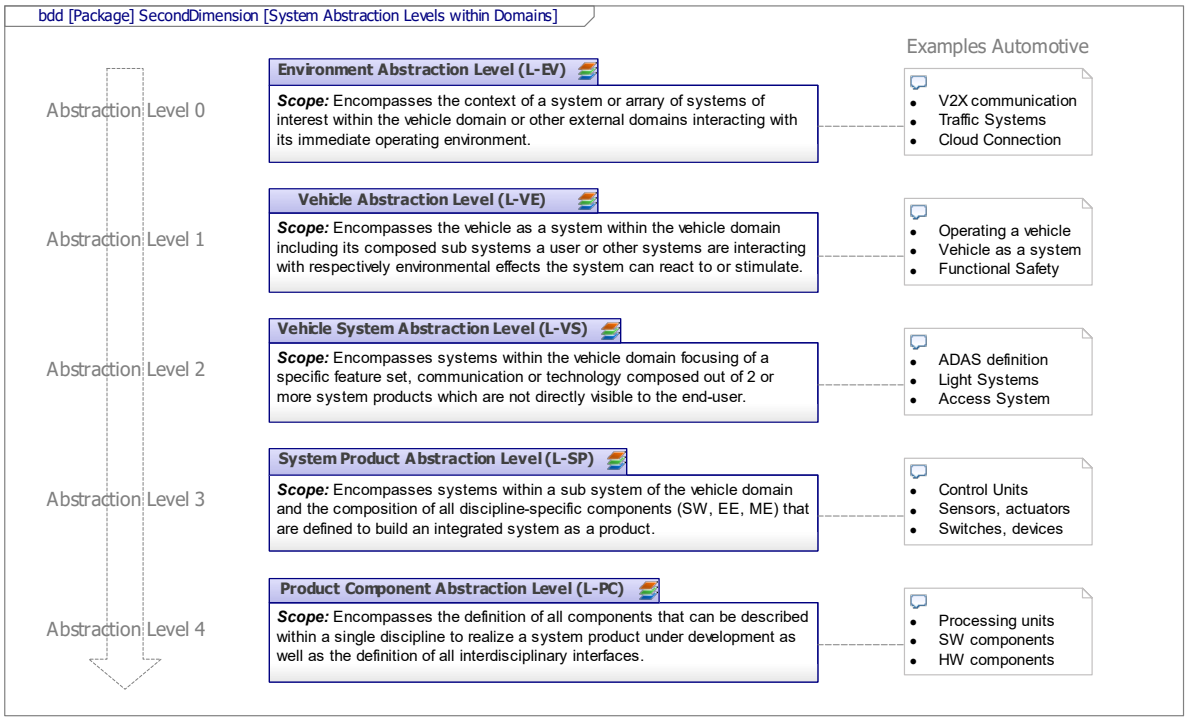


図4: 車両ドメインにおける5つの適用された抽象化レベルの例

7 階層の深さ

システムが抽象化レベルで分析され、すべてのビューポイントにわたってモデル化される場合、これを完全なアーキテクチャステップと呼びます。このプロセス内では、システムや機能ブロックが異なるビューポイントにおいて異なる粒度を示すことがあります。したがって、階層の深さは、特定のビューポイントの実際のシステム抽象化レベル内でシステムの分解の粒度を定義します。図5に示すように、これによって各ビューポイントにおけるシステムの詳細度が決まります。

最小で推奨される階層の深さは1であり、これによりシステムの分解を簡潔に保つことができます。しかし、重要な点として、階層の深さは、同じシステム抽象化レベル内で各ビューポイント間で一貫している必要はありません。これは、各ビューポイントに適用される分解スキームによって分解の粒度が異なるため、異なる階層の深さが生じる可能性があるためです。

標準化されたシステム分解を確保するために、iSEFはシステムや機能を互換性があり再利用可能なビルディングブロックとして分解するための指針となる、参照分解モデルのセットを提供しています。

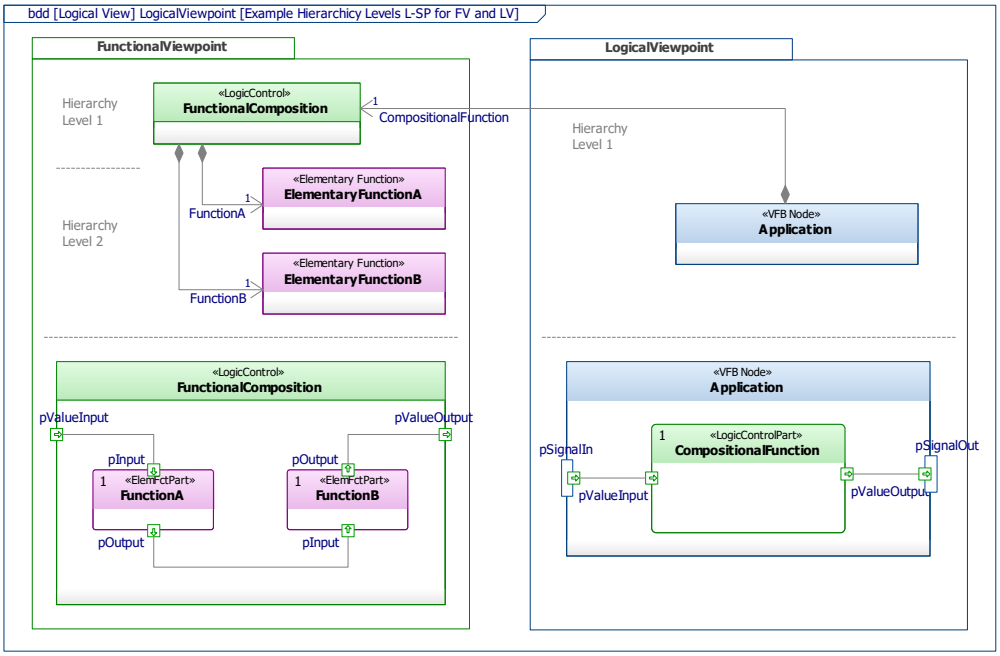


Figure 5: Example of a Hierarchy Depth of "2" in the FV and "1" in the LV at the L-PC

8 モデルビューポイント間の要素のリンク

異なるビューポイント間で一貫したアーキテクチャ記述を確保するためには、同じモデリングビューポイント内のアーキテクチャ要素と、異なる視点間の要素間にリンクパターンを確立する必要があります。iSEFメソッドは、拡張されたSysMLセマンティクスを導入することでこのニーズに対応しています。このフレームワーク内では、さまざまな視点からの要素が相互に結びつき、機能要素が直接論理要素に統合され、さらに技術要素に構成されます。これらの技術要素は、物理的なコンポーネントにさらに組み込まれます。

図6に示されるように、関与するすべてのアーキテクチャ要素のリンクは、それらの相互接続されたインターフェースによって強化され、モデルチェッカーによって整合性がチェックされます。整合性は、プロキシポート（Proxy Port）がフルポート（Full Port）にインターフェース定義を通じて埋め込まれるネストポートモデリングによって維持され、論理信号が1つ以上の機能値をカプセル化します。この信号はテクニカル・インターフェース（技術インターフェース）を通じて伝達され、技術ラインに沿って運ばれ、フィジカル・インターフェース（物理インターフェース）に接続されます。このフィジカル・インターフェースは、すべてのビューポイント間の全体的な接続を実現するためのソリューションを表します。

技術的および物理的な制約がこのモデリングステップで関与し、物理アーキテクチャはテクニカル・ビューポイント（技術視点， 技術しんてん）からの技術的な決定に基づいています。さらに、テクニカルモデリングは、ロジカル・ビューポイントからのアーキテクチャ選択に依存します。ロジカルモデリングビューポイントは、ファンクショナル・ビューポイントで作成された特定の機能設計に基づいた実装を反映しています。また、ファンクショナル・ビューポイントの機能設計は、オペレーショナル・ビューポイントで概説された活動やユースケースを評価した運用分析の結果と見なされます。

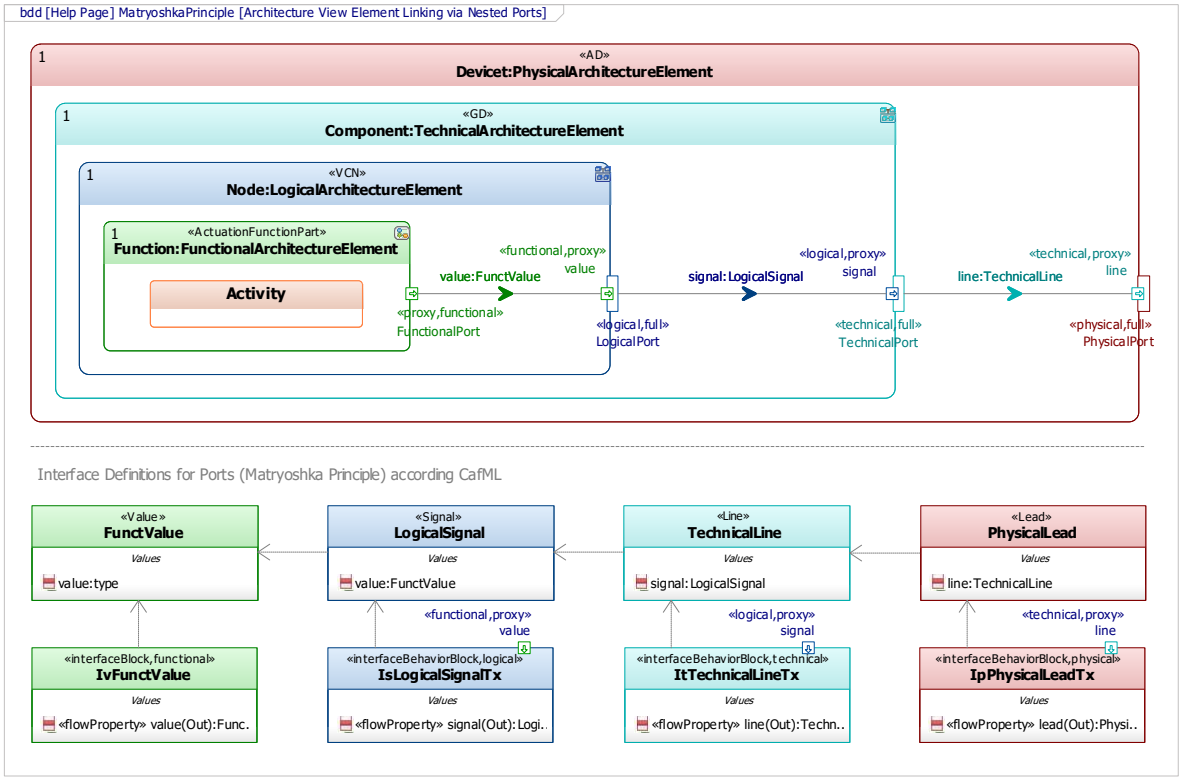


Figure 6: Linking Elements and Interfaces Across Successive Viewpoints

図6では、SysMLに加えて導入されたiSefMLのステレオタイプであるインターフェースビヘイビアブロック（InterfaceBehaviorBlock）の使用も示されています。この特定のインターフェースブロックは、要素間の接続と、ビューポイントを超えた情報の正式な転送を、静的および動的（シミュレーションによる）に促進します。このアプローチにより、ISO 26262（2018）などの最高の安全性整合性レベルに準拠したセミフォーマルなアーキテクチャおよび設計が作成可能となります。Rhapsodyのようなモデリングツールは、これらの原則を適用して記法とセマンティック検証を自動化し、機械によるモデル検証を実現します。この原則は、異なる抽象化レベルとビューポイント全体で、すべてのアーキテクチャ要素をシームレスに接続するために重要です。

一方で、iSEFは構成されたパーツの一貫したリンクを可能にし、モデル内のすべての要素が他のビューポイントに対応する要素と論理的に接続されることを保証します。これにより、システムアーキテクチャの整合性が維持され、システムが機能、論理、技術、物理コンポーネントに明確かつ構造的に分解されることが促進されます。また、このビューポイント間リンクアプローチにより、アーキテクチャ要素がパズルのピースのように適合し、一つのビューポイントの要素間インターフェースが次のビューポイントのものと完全に一致するようにします。この整合性により、あるアーキテクチャビューから次のビューに移行する際、要素が無秩序に配置されるのではなく、意図的に設計され、シームレスに統合されることが確実になります。

ビューポイントとシステム分解の関連に関するさらなる洞察は、コンチネンタルによる以前の出版物（Seidel & Forlingieri, 2023）で確認できます。

注釈

著者は、「invenio SEチーム」と協力して、ビューリンクの原則に直接適用できる一連の事前定義されたシグナルライブラリを積極的に開発しています。これらのライブラリは現在、自動車分野向けに設計されていますが、将来的には他の産業にも適用範囲が拡大される予定です。これらのライブラリによって提供される事前定義されたインターフェースは、効率的なモデリングの基盤を形成しており、iSEFプロファイルでは、数回のマウスクリックで即座に使用できるシグナル、サービス、テクニカルライン、または物理的なリード定義を実装することが可能です。

この効率化されたプロセスにより、標準化された機能チェーン、論理構造、技術的ソリューション、物理的な実装を迅速に開発することができます。また、ソフトウェア定義の車両アーキテクチャも、このiSEFフレームワークの恩恵を受けており、システムアーキテクチャからソフトウェアアーキテクチャへのシームレスな派生がより実用的かつ効率的になります。さらに、iSEFフレームワークは、共通の事前定義されたインターフェースに基づく標準化されたアーキテクチャの開発をサポートすることで、OEMとサプライヤー間の協力を促進します。この標準化により、自動車のサプライチェーンにおけるさまざまな組織間で、より統合された一貫した開発プロセスが促進されます。

9 システム設計のための体系的モデリングステップ

システムアーキテクチャの構築原則の簡単な導入に続き、先に紹介した7つのモデルビューポイントを含む、著者はACMBSEを使用したシステムモデリングを、iSEFを使用したシステムモデルの開発を示すワークフローを通じて実演しています。次のセクションで紹介される例は、実際の開発範囲に基づいた車両の外部照明機能の設計と実装に焦点を当てています。簡潔にするために、この論文では全体の範囲の一部のみを取り上げています。

著者は、分析から設計までの8つの基本的なモデリングステップを以下のように示しています：

1. **バリアビリティ・ビューポイント（VV）を使用したフィーチャー定義**：機能および技術的なフィーチャーとその変動性に基づく関係を定義し、バリエーションポイントを通じて管理します。
2. **オペレーショナル・ビューポイント（OV）を使用したシステムコンテキストと対象システム（SoI）の定義**：これは、外部要素（アクター）との構成および相互作用を含みます。
3. **リクワイアメンツ・ビューポイント（RV）を使用した要求事項の導出**：このステップでは、システム要素、設計制約、インターフェース、および他の設計側面のトレーサビリティを含みます。
4. **オペレーショナル・ビューポイント（OV）を使用した動作分析**：これは、ユースケース分析や再帰的なシステム活動の精緻化を含む、期待されるシステム動作の分析に焦点を当てます。
5. **ファンクショナル・ビューポイント（FV）を使用した機能設計の開発**：状態遷移図の定義、活動の精緻化、および完全な機能構成の達成を含みます。
6. **ロジカル・ビューポイント（LV）を使用した論理コンポーネントの開発**：特定のターゲットシステムのための構造的システム構成とインターフェース定義を作成し、ポートリンクを行います。
7. **テクニカル・ビューポイント（TV）を使用したシステムの技術的ソリューションの定義**：問題空間とソリューション空間を橋渡しするために、論理アーキテクチャ要素を統合します。
8. **エンドユーザーシステムの物理アーキテクチャの実現**：既存または新たに定義された実装可能な物理コンポーネントやアセンブリを使用し、物理インターフェースの定義を含みます。

以下のモデルは、架空のボディコントロールユニット（BCU）の開発を簡略化します。このプロセスは、車両レベルから始まり、必要なシステム機能を特定し、システム機能とインターフェースを設計し、製品レベルで機能やコンポーネントを開発します。最終的に、これらのコンポーネントは物理的な電子制御ユニット（ECU）に統合され、車両の照明機能を含む一連の機能を実装します。これにより、ACMBSE手法の適用を示す具体例となります。

9.1 ステップ1 - システムのフィーチャー定義

開発者がシステムアーキテクチャを定義し始める前に、エンドユーザーのニーズを満たすためにシステムやシステム製品ラインが備えるべき重要な属性を明確に理解しておくことが重要です。これには、ForlingieriおよびWeilkiens（2022）によって記述されたモデリングアプローチに従い、バリアビリティ・ビューポイント（VV）を活用して必要な機能的および技術的フィーチャーを分析することが含まれます。

この理解に基づいて、機能的フィーチャーは、システム・オブ・インタレスト（SoI）を機能的な観点から特徴づけ、システム設計や動作に対する深い理解を持たないエンドユーザーやステークホルダーにも理解できるようにします。機能的フィーチャーは、SoIのニーズやその能力の総合的な表現を示し、機能要件の導出の基礎として機能します。これは、Beuche et al.（2004）やForlingieri（2022）でも示されています。

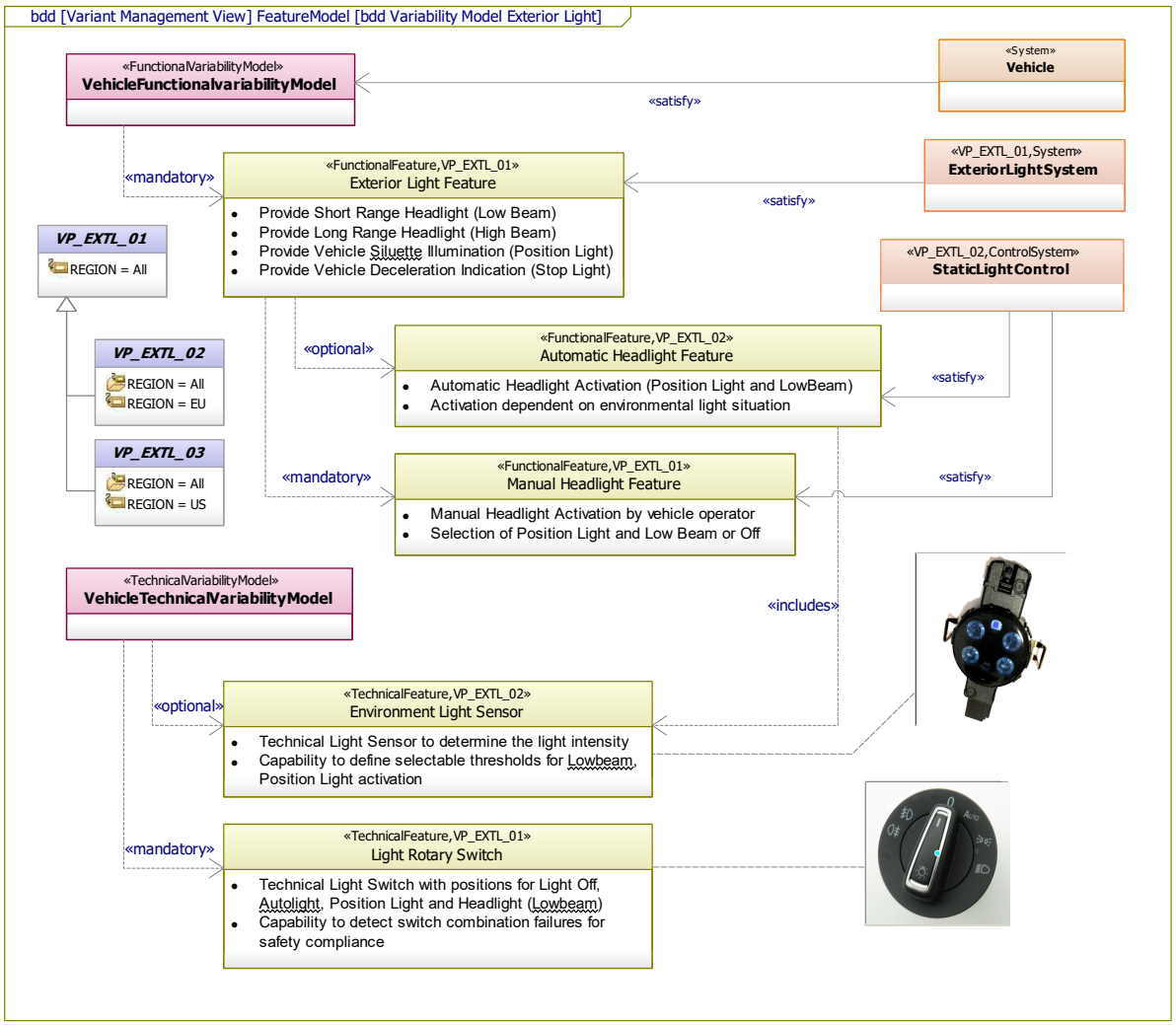


図7: 機能的および技術的フィーチャーを用いたバリアビリティモデルのモデリング

テクニカル・フィーチャー（Technical Features）は、システム・オブ・インタレスト（SoI）を技術的な観点から性能や技術に関連して特徴づけ、実装の基礎的な部分に対する深い理解を持たないエンドユーザーやステークホルダーにも理解できるようにします。これらは、求められるSoIに対する技術的制約、性能能力、または技術的限界の総合的な表現を示し、非機能的な技術要求の導出の基礎となります。テクニカル・フィーチャーは、技術的な境界条件と機能的フィーチャー（Functional Features）との依存関係に基づいて、システムの特性を制約することがよくあります。この依存関係は、例えば図7で「includes」関係を通じて示されています（Beuche et al., 2004; Forlingieri 2022）。

機能的フィーチャー（Functional Features）およびテクニカル・フィーチャー（Technical Features）は、図7に示されるように「必須（mandatory）」フィーチャー、「オプション（optional）」フィーチャー、または「代替（alternative）」フィーチャーといった依存関係によって特徴づけられ、他のフィーチャーとの関係（例: 「排他的（exclusive）」または「包括的（inclusive）」）としてモデル化することができます。例えば、図7では「必須」の「手動ヘッドライト機能（Manual Headlight Feature）」とオプションの「自動ヘッドライト機能（Automatic Headlight Feature）」が示され、どちらも「静的ライト制御（Static Light Control）」というユーザーが認識できる機能を通じて実現されています。また、「環境光センサー（Environment Light Sensor）」を含むテクニカル・フィーチャーは、光の強さを測定するために使用され、与えられた機能的フィーチャーとテクニカル・フィーチャーの間の「包括的」依存関係を示しています。

フィーチャー階層内の任意のバリエーションポイント（例: 「VPA-EXTL-02」）で定義されたバリエーションポイントは、バリアントに対応することができます。これらのバリエーションポイントは、図の「静的ライト制御」機能ブロックに示されているように、アーキテクチャ要素のバリアントの関連性を明確にするために、任意のアーキテクチャ成果物に追加できます。モデリングツールと提供されるプロファイルを組み合わせることで、モデルの妥当性チェックと、Rhapsodyのようなモデリングツールの図ビューに適用できるフィーチャー特有のコンテンツの可視化が可能になります。

9.2 ステップ2 - システムおよびコンテキスト定義

iSEFメソッドにおいて、システムアーキテクチャの開発の次のステップは、システム・オブ・インタレスト（SoI）を含むシステムコンテキストを定義することです。これは、オペレーショナル・ブロック定義図（OBDD）で関連するシステム要素をモデリングし、すべてのシステムコンテキストアクターを特定することで達成されます。

次のステップでは、1つまたは複数のオペレーショナル内部ブロック図（OIBD）を使用して、SoIとコンテキスト要素（アクター）間の相互作用を特定します。図8に示されているように、「運転車両（Operating Vehicle）」がSoIとしてモデル化されています。灰色の要素は、この定義におけるコンテキスト要素を表し、図8に示されています。環境レベルでの抽象的なオペレーショナルイベントを特定することが重要です。図8では、照明システムのためのオペレーショナルイベントを使用したコンテキスト定義が示されています。

SoIの構築の基礎となるテクニカル・フィーチャーは、SoIの理解においても非常に重要であり、これらはシステムおよびサブシステムの開発の後続のステップに制約を与え、後に物理的に実現される必要があるインターフェースを定義します。事前定義されたオペレーショナルイベントは、次のオペレーショナル分析（図8参照）に使用され、SoIとコンテキスト要素間のユーザー相互作用を特定し、ユースケースの一貫した定義を容易にします。もし、以前のアーキテクチャステップからインターフェースが生じ、現在のシステム抽象

化レベルの要件の一部となっている場合、それらは技術的および論理的インターフェースとしてアーキテクチャに組み込むことができます。

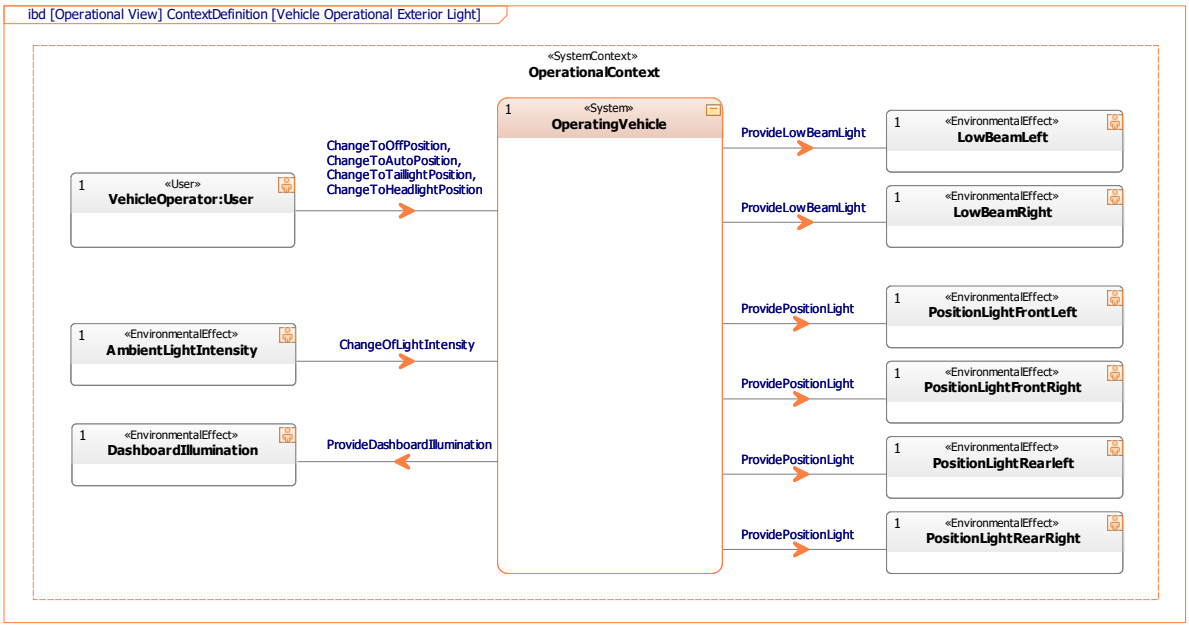


図8: コンテキスト定義と機能的照明システムの識別

9.3 ステップ 3 - 再帰的要求工学

iSEF メソッドは、ACMBSE の原則と組み合わせて、Zig-Zag メソッド（Weilkiens 2008）を再帰的に使用してシステム要求を作成します。図8は、システムの関連する抽象レベルにおける機能、要求、およびアーキテクチャ要素のトレース可能性を示しています（Seidel & Forlingieri 2023）。

Zig-Zag メソッドの適用は、図9に示された「Z」をたどることによって行われます。初期のシステム抽象レベル（Lx）では、ブラックボックスシステムの範囲を反映する要求が作成されます。例えば、「Vehicle」システムが「ExteriorLight」要求を満たす例です。次に、アーキテクチャのステップが実施され、「Vehicle」システムをより小さな部品やサブシステム（図9の「Static Light Control」など）に分解します。アーキテクチャ上の決定は、また「Vehicle」システムの要求から洗練された根拠に記録されます。これらの根拠は、図9に示すように、サブシステム（例：StaticLightControl）の次のシステム抽象レベル（Lx+1）で要求を導出するための基盤を形成します。この要求仕様とアーキテクチャプロセスの交互の実施は、サブシステムが単一の分野（例：電気/電子またはソフトウェア）で実装できるようになるまで、すべてのシステム抽象レベルにわたって繰り返されます。

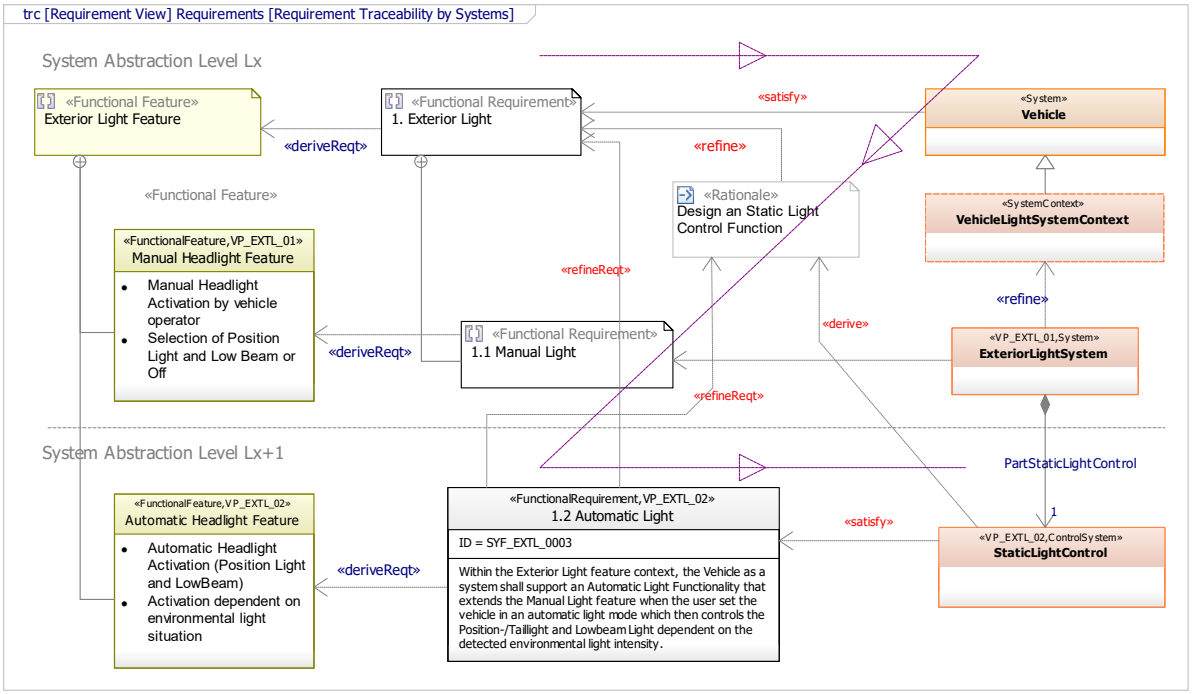


図9: アーキテクチャ要素と要求のトレーサビリティ

9.4 ステップ 4 - システムの運用分析

システムの運用分析を行うことは、システム・オブ・インタレスト（SoI）の活動や状態の振る舞いを特定する上で重要なステップです。ユースケース図は、機能的要求事項および基礎となるシステム活動から派生したユースケースの洗練を支援します。これらの活動は、運用分析のさまざまな抽象レベルでさらに詳細に精練することができます。iSEFガイドは、システムプロセス（System Process）、システムユースケース（System Use Case）、セカンダリーユースケース（Secondary Use Case）、および連続ユースケース（Continuous Use Case）など、専門的なメタクラスを提供します（Wilkins 2008）。

ユースケース分析は、サブシステムがまだ特定されていない場合や、システムの分解方法が不明な場合に特に有用です。しかし、システムの分解が詳細な要求事項や技術的制約により事前に定義されている場合、ユースケース分析はさらに詳細な情報を展開する最も効果的なアプローチではないことがあります。そのような場合には、SoIを既に特定されたサブシステムに直接分解し、次のシステム抽象レベルで運用分析を実施する方が実用的です。

この方法論の特長として、図10の例では、以前に定義された運用イベントを再利用して、システムユースケースとともに与えられたコンテキスト要素のイベントを一貫してモデル化しています。基盤となるシステム活動、例えば「ロービームの起動（Low Beam Activation）」は、詳細な振る舞いの精練のためのさらなる分析ステップを示し、状態機械の作成を可能にします。

iSEFを使用してモデル化することで、活動や状態機械の意図された振る舞いを簡略化してシミュレーションすることができます。これにより、システムの詳細を早期にステークホルダーと議論することができ、不必要な開発サイクルを回避する助けとなります。

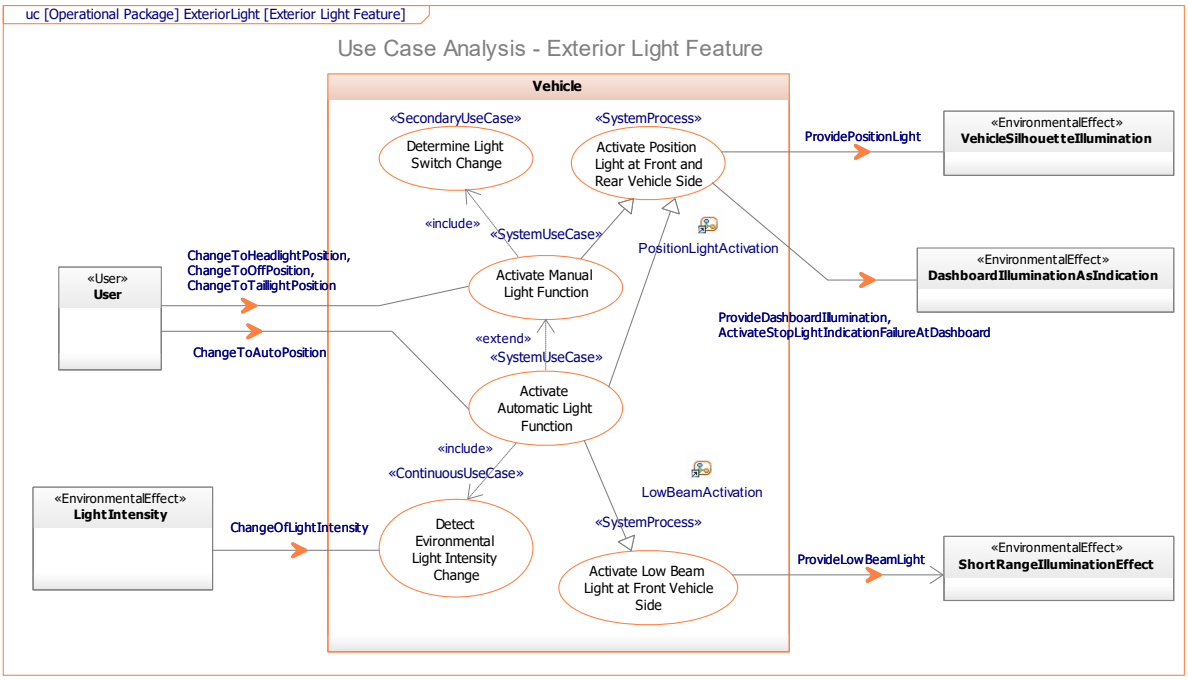


図10: 基盤となるシステム活動を用いた運用分析

9.5 ステップ 5 - 詳細機能設計

Harmonyの原則（Douglas 2017）とiSEFの追加拡張を指針として、前の運用分析ステップで策定されたシステム活動は、Essential Activities（基本活動）やNode Activities（ノード活動）に精緻化され、同じ抽象レベルまたは一つ下のレベルの機能ブロック要素（例：Logic Control機能ブロック）に配置されます（ユースケース分析の詳細については後のホワイトペーパーでさらに説明します）。呼び出し側のSystem Activity（システム活動）でモデル化されたスイムレーンと機能ブロックは、運用ビューの要素と機能要素とのリンクを促進します。

機能設計ステップでシステム活動を実行可能なノード活動やState Machines（状態機械）に精緻化する際、運用ビューの運用イベントは通常、SysMLのプロキシポートを使用して情報フローに変換されます。これらのインターフェース定義（例：「IAutoLightRequest」）は、機能ブロック（例：「Headlight Control」）のポートを通じて実装されます。ほとんどの組み込みシステムには状態機械が含まれているため、図11の例ではこの概念が説明されます。状態機械は、以前に定義された活動やイベントを精緻化し、ターゲットとなるデータ要素、操作、およびインターフェースを利用しシミュレーションします。具体的には、図8の「LightRequest」を評価します。すべてのデータ要素と動作が正確に設計されると、システムをシミュレーションして機能ブロックの意図された動作を検証することができます。

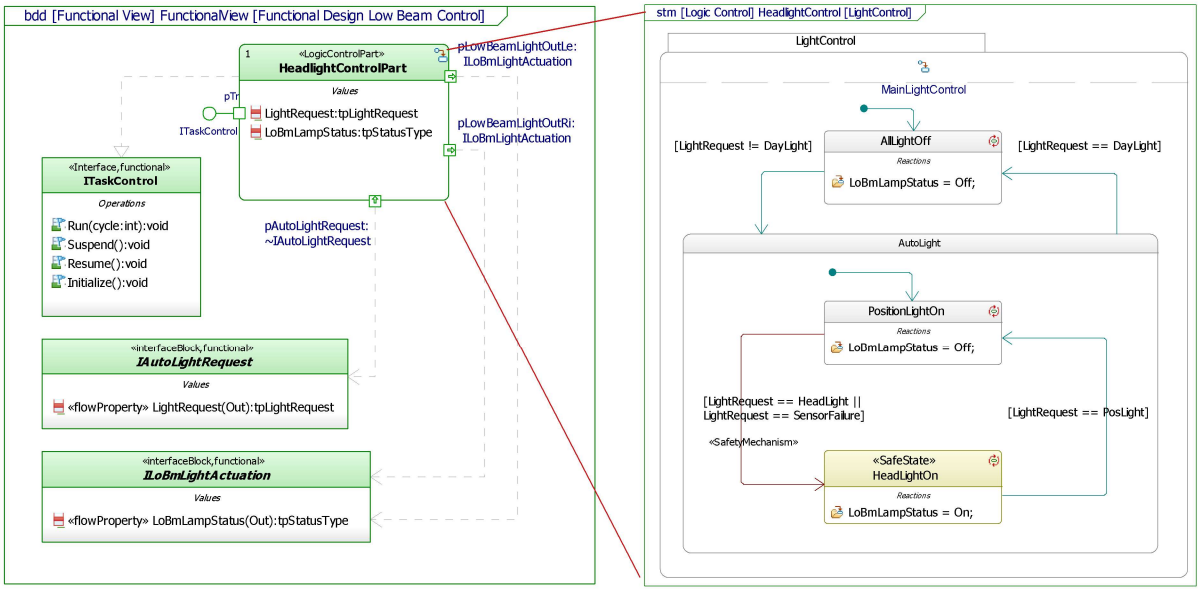


図 11: ロービームライト制御の機能設計

機能的な観点から、多くのiSEFで開発される機能は高い再利用性を考慮して設計されています。これらの機能は、機能要素間の通信のためにポートを使用することで、機能ブロックの独立性を確保することによって実現されます。図11で示された「ITask Control」インターフェースの側面に加えて、機能的なビューでは、機能値の交換と操作のサポートのための機能呼び出しを実装するための事前定義されたインターフェースが強調されます（API）。これらのAPIは、後に論理ビューからサービス指向のインターフェースに統合されます。

9.6 ステップ 6 - 論理コンポーネントの設計

論理モデルの観点からの主要な目標は、機能要素を論理ノードとして知られる論理ユニットに整理し、これらのノードをより大きな論理システムに統合することです。これらのノードは追加の明示的に実装された動作を含まず、すべての動作モデルはすでに機能要素内で開発されています。図 12 は、他の機能ブロックを制御するが追加の機能ユーザー動作には寄与しない「アプリケーション管理」コンポーネントの統合例を示しています。

機能要素は通常、ポートを使用してノード内で他の機能と通信できます。このアプローチは必須ではありませんが、機能要素の柔軟な再編成を可能にし、独立して開発できることで設計の再利用と標準化を最大化します。

電子制御ユニット（ECU）内の機能は、異なるオペレーティングシステムやソフトウェアフレームワーク（例：AUTOSAR Classic (2022)）で動作する際に追加の制約を実装する必要があります。ただし、これらの機能はさまざまなターゲットシステムに適用できるように設計されるべきです。適応は、論理ノード内で論理ポートを使用して処理され、これは「モデル視点間の要素のリンク」の章で説明された原則に従います。

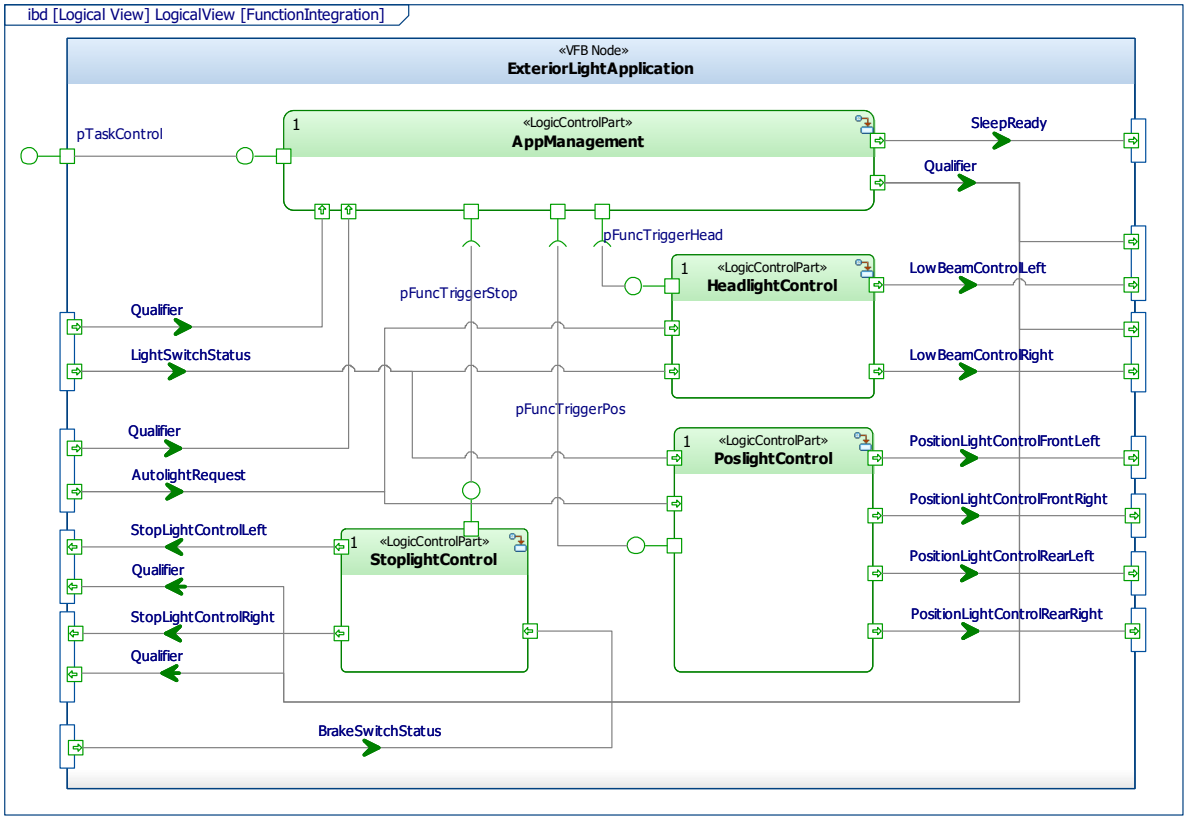


図12: 仮想ファンクションバスノードへの機能要素の統合

フルポート（Full-Port） は、インターフェースビヘイビアブロック（Interface Behavior Blocks）によって特徴付けられ、機能値の内部変換を外部の標準化されたインターフェース定義またはカスタム信号・サービスインターフェース定義と行います。この原則により、SimulinkブロックやMatlabモデル（IBM Docu 8.3）や、論理インターフェースに直接対応しないサードパーティの機能設計を論理ノードに柔軟に統合することができます。論理ノードとそのインターフェースは外部から見ると同じに見えますが、内部の機能実装は異なる場合があります。図12は、例えば「ライトスイッチの状態」や「修飾子」に関する情報を含む論理信号ポートが「ヘッドライト制御」と機能的にリンクし、それが「左/右低 beam コントロール」の機能値を介して論理出力ポートに接続される様子を示しています。

ACMBSEでは、機能ビューから論理ビューへの要素リンク時にアロケーション（allocations）は通常避けられます。オブジェクト指向プログラミングのように、機能ブロックは論理ノード内でインスタンス化され、機能インターフェースが信号ポートに接続されます。内部機能ポートは、図12で示されているように、機能要素へのネストされたインターフェースを明らかにします。

このモデル化アプローチは、ISO 26262規格（ISO 1 2018）への準拠が要求される安全クリティカルなアプリケーションにおいて重要な、高度に一貫した完全に実行可能なアーキテクチャモデルを実現します。iSEFガイドラインが正しく適用されると、異なるモデルビュー間の要素リンクは同じ抽象レベル内の要素に制限され、「レベルホッピング」による不正確で実装不可能な統合を防ぐことができます。iSefMLプロファイル内の事前定義された分解モデルと専門的なステレオタイプは、アーキテクチャ要素の設計中にリアルタイムでモデルの検証を容易にするために提供されています。

9.7 ステップ 7 - 技術的システム開発

システム・オブ・インタレスト（SoI）の技術的視点は、ソフトウェアおよびハードウェア分野のコンポーネントを統合することでモデル化されます。技術的視点は、システムの論理ノード（アプリケーションなど）とマイクロコントローラのような技術的要素との相互作用を主に提供します。

図13に示すように、論理ノードが外部要素（例: ドライバノード）とどのように通信するかを、処理ユニットの技術的インターフェース（ポート）を通じて確認することが重要です。

また、マイクロコントローラの各技術的インターフェースは、アナログまたはデジタル入力や、コントローラエリアネットワーク（CAN）バスによる通信のためのデータインターフェースを含むハードウェア-ソフトウェアインターフェース（HSI）を表します。既存のコントローラと直接統合するのではなく、このモデルビューはコントローラ構築のために技術的に重要なインターフェースを特定することに焦点を当てています。実際のコントローラの実装は、物理モデルビューでの後のフェーズで行われます。

技術モデルビューについては、ここでは論理ノード（ソフトウェアコンポーネント）が統合されていますが、実際にはコード生成とコンパイルプロセスが実行される必要があります。しかし、これらのモデルビューは、UMLプロファイルを使用して展開図として表現され、ここではシステム表現とは無関係なソフトウェア固有の技術的および物理的アーティファクトに焦点が移ります。

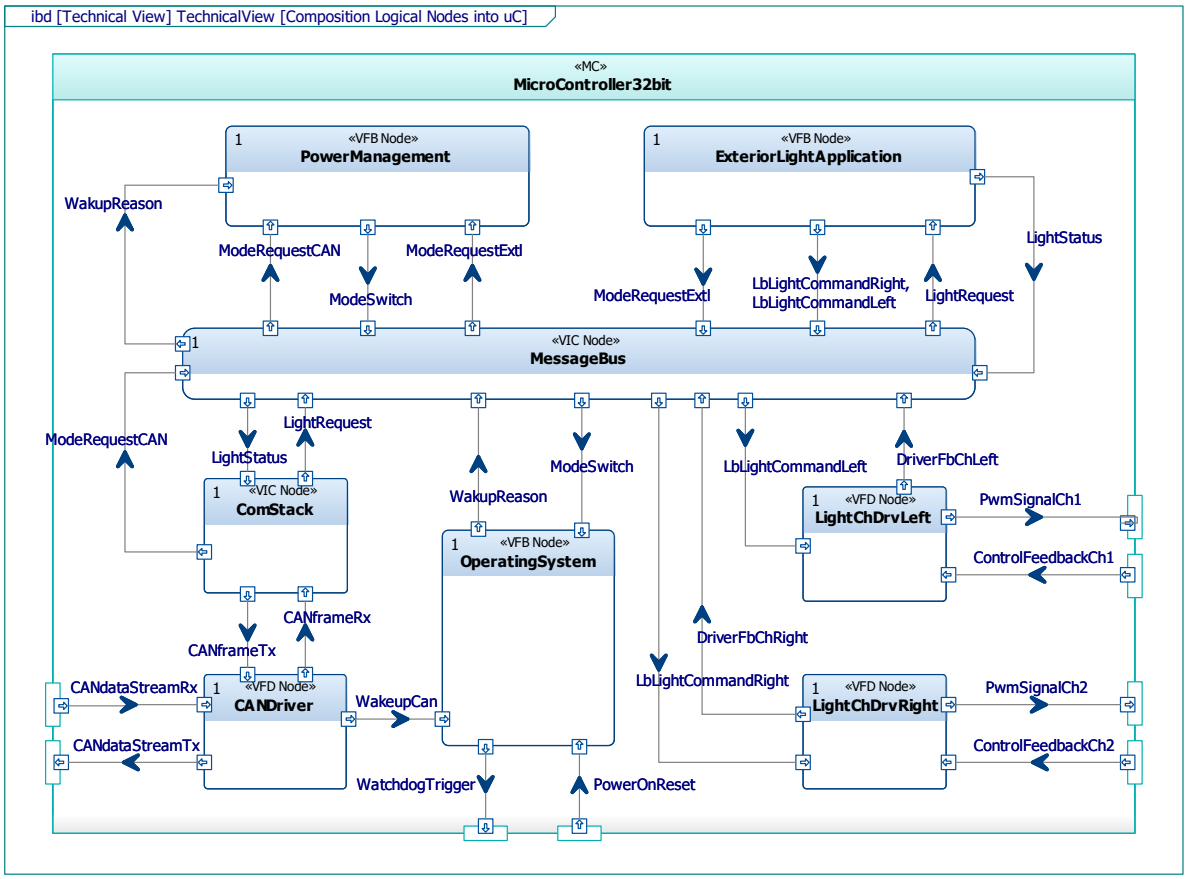


図13: 技術的マイクロコントローラーにおける論理ノードの実装

iSEF手法に従った技術アーキテクチャのモデル化、特にHSI要素を用いることで、ソフトウェア設計前にプロセッサ内の論理要素を実行可能にシミュレーションすることが可能になります。これにより、開発段階でスケジューリング、リソース、およびタイミングなどの技術的制約を早期に検証することができます。図13は、図12の「外部照明アプリケーション」がマイクロコントローラーアーキテクチャに統合され、「エネルギー管理」や「通信スタック」などの他の論理ノードとどのように相互作用するかを示しています。さらに、すべての論理信号をルーティングするためにメッセージバスが統合されています。

このアプローチにより、ハードウェアとソフトウェア間のハードウェア依存の安全メカニズムのシミュレーションが可能になり、ASIL D（ISO 26262, 2018）までの安全クリティカルなシステム開発を検証できます。

iSEF手法により、実際のハードウェアが利用可能になる前にシステムプロトタイプを作成でき、開発コストを大幅に削減できます。しかし、このような詳細なモデル化はアーキテクトの知識を試し、すべての機能ノードとインターフェース要素を完全に開発するための規律あるアプローチが必要です。したがって、モデリングの範囲には妥協が必要であり、理解とシミュレーションに必要な部分に焦点を当て、他の部分は静的で完全には実行可能でないままとなることがよくあります。

また、モデルベースのモデリングアプローチには、システムエンジニアが目標を達成するための適切なトレーニングが必要です。そのため、invenio Systems SEでは、フレームワークとライブラリを効果的に使用するために必要なトレーニングを実施しています。

9.8 ステップ 8 - アーキテクチャの物理的実現

物理的モデリングは、システムおよびハードウェアエンジニアが最終製品、主に電子制御ユニット（ECU）のアーキテクチャを設計するのに役立ちます。ただし、物理的モデリングは半導体チップセットの設計、アーキテクチャの文書化、または車両のE/Eシステム全体のモデル化にも適用できます。iSEFはこの目標をさまざまな方法でサポートします。事前定義されたテンプレートおよびポート構成に基づいてマイクロコントローラを開発するためのモデル要素を提供します。また、ソフトウェアおよび基本システムのためのデバイスを設定し、エンドユーザーシステム（たとえば車両）を正確に定義する手助けをします。たとえば、すべての制御ユニットのモデルと統合された配線ハーネス全体が統合されます。

これにより、OEMは提供されたモデルコンポーネントを使用して非常に複雑なシステムを統合および検証でき、単一のプロトタイプを構築する必要がありません。これにより、通常、修正ループが少なくなり、開発コストの大幅な節約が可能になります。

図14は、技術的要素によって定義された技術的解決策に基づいてECUのハードウェアアーキテクチャを構築するためのアプローチの一例を示しています。この例では、以前に設計された技術的要素（たとえば、図13の技術的マイクロコントローラ）が物理的に開発されるECUに統合され、ECUの必要なポートに接続されます。このアーキテクチャにより、システムエンジニアはすべての技術的解決策と必要な物理的制約をハードウェアエンジニアに渡すことができ、ハードウェアエンジニアはこれらの制約を考慮しながら物理的システム（ECU）を設計します。

iSEFは、将来的にこれらのモデリングアプローチをサポートし、開発プロセスを加速するために、多くの事前定義された物理的標準コンポーネントを提供することを目指しています。

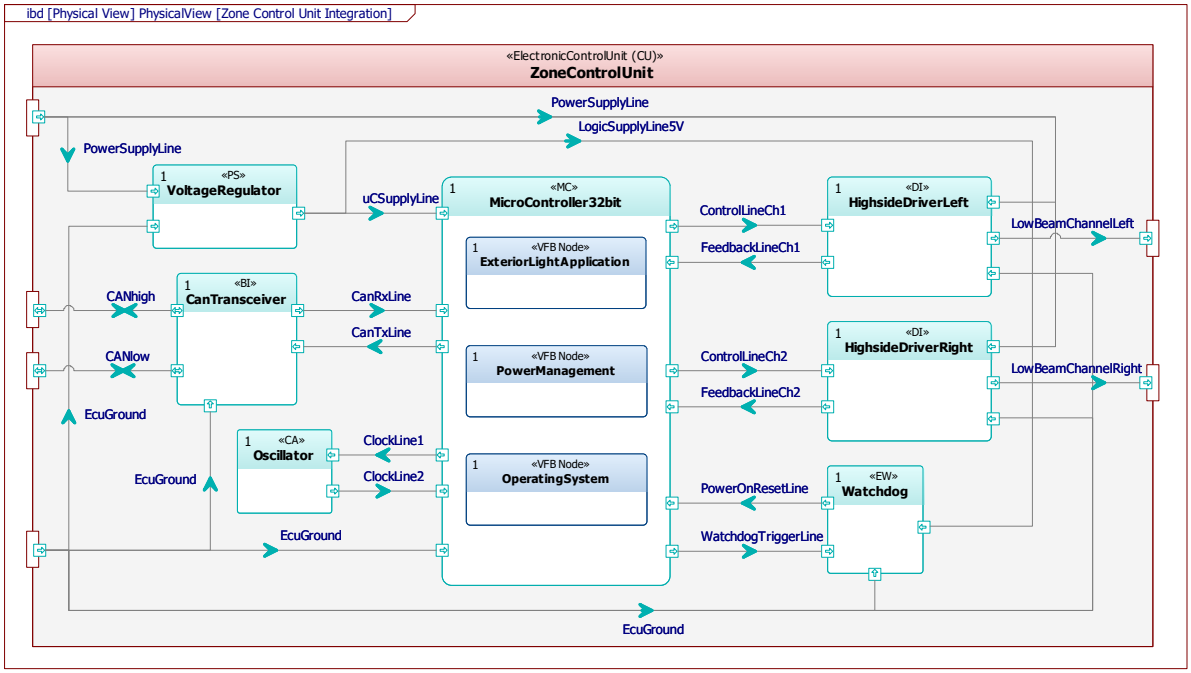


図14: 物理制御ユニットにおける技術要素の構成

別のアプローチとして、この記事では扱われていない、最終製品の完全な物理アーキテクチャをモデル化する方法があります。このアプローチでは、ECUがすべての関連物理要素を統合して、完全に最適化されたハードウェアアーキテクチャを作成します。この最適化は、技術的特徴や与えられた技術的制約に基づいて性能とコストを考慮するもので、結果として具体的なハードウェア設計が得られます。このようなアーキテクチャ、いわゆる物理ブロックダイアグラムは、再利用可能で標準化された物理サブコンポーネントの開発を可能にします。これにより、解決策が既知で、モジュール要素から建築的に組み立てることができる場合には、開発時間を大幅に短縮できます。このアプローチは、特に自動車製造の大規模プラットフォームにおいて有利であり、開発を大幅に効率化することができます。ただし、これらのモジュール要素を作成するためには、初期投資が大きく必要です。

10 結論と展望

今後数年で、車両電子機器における大きな変化が、システム開発のためのスケーラブルで柔軟なプラットフォームの必要性を高め、次世代のE/Eアーキテクチャ（E/E　アーキテクチャ）への移行が求められるようになる（Seidel & Forlingieri 2023）。この新しい挑戦的な文脈に対応するためには、現在のルールや伝統的な文書ベースの方法を超えるシステムアーキテクチャ開発アプローチが必要である。本記事で紹介されているACMBSEアプローチは、invenio Systems Engineering Framework（インヴェニオ・システムズ・エンジニアリング・フレームワーク）に実装され、モデルベースの開発におけるシステムアーキテクチャの中心的役割を強調している。著者は、このアーキテクチャアプローチの構築と実装の重要性をいくつかの主要な指針を通じて強調している：

(1) iSEFのようなモデリングフレームワークを開発することは、さまざまなエンジニアリング部門における一貫した効率的な開発にとって不可欠である。この実践は、AirbusがMOFLT（MOFLT）を採用しているように、他の産業にも適用できる。ACMBSEの原則を適用することで、アーキテクチャの理解、コンポーネントの定義、チーム内およびステークホルダーとの持続可能なコミュニケーションの改善などの課題を克服することができる。

(2) ホワイトペーパーは、システム開発におけるさまざまなが相互に接続されたモデルの視点を確立する重要性を強調している。iSEFフレームワークは、類似のフレームワーク（Roques 2016; Ducamp et al., 2022）と同様に、異なる視点からシステム設計を可能にし、システムからソフトウェアアーキテクチャへのシームレスな統合を提供する7つの基本的なモデルの視点を提供する。この方法は、車両統合からECU開発までのすべての重要なアーキテクチャの側面を網羅し、全体の製品説明の橋渡しとして機能する。

(3) 著者は、iSEFと導入された視点の多くの潜在的なアプリケーションのうちの1つを体系的に示した。MBSEの基本的な原則は、この記事で示されたビューのリンクの原則を適用することでiSEFで確立することができる。すべての視点とシステム抽象レベルが常に同時に必要とされるわけではないが、複雑なシステム（車両など）の開発中に同じ視点の異なる側面を利用することができる。

このホワイトペーパーは、ACMBSEを使用したシステム開発の分析および設計フェーズに主に焦点を当てているが、8つの簡単なステップでシステムアーキテクチャのモデリングを示している。しかし、1つの重要な側面はモデルベースの検証とバリデーションである。複雑なシステムアーキテクチャの精度と一貫性を確保するには、モデルの意味的および構文的な正確性、そして相互作用するモデルコンポーネント間の一貫性を検証する必要がある。そのため、将来のツール、iSEFを含む、はフレームワークの基本的な方法や検証メカニズムを実装し、ユーザーが容易に適用できるようにするための技術的な手順（例：ジャストインタイムモデルチェック）を組み込むべきである。

著者によって簡単に言及されたもう1つの重要な点は、機能的および技術的特徴の変動性のモデリングであり、製品ライン開発における望ましい特性を最初から考慮するものである。しかし、この記事の目的は、これらの特徴をどのように体系的にモデル化し、複数の製品バリアントを開発するために使用するかを示すことではなかった。ForlingieriおよびWeilkiens（2022）の例に倣って、将来の記事では、iSEFを使用してさまざまな視点や抽象レベルでバリアントモデリングをどのように実現できるかを示すことができる。

結論として、この記事は、モデルツール内で専ら表現されたアーキテクチャ中心のモデルベースシステム開発の側面を強調している。しかし、システムモデルをエンジニアリングライフサイクル内の他のドメイン特化型の分野と接続するための橋渡しとして使用する重要性を強調することが重要である。MBSE（モデルベースシステムエンジニアリング）は、要求管理、テスト管理、ソフトウェア開発、または機械的分野を置き換えるものではない。むしろ、iSEFは、これらの分野のシームレスな統合を促進し、一貫したシステム開発プロセスを実現するための包括的なモデルを通じて、それを可能にする。invenio Systems Engineering Framework（iSEF）は、アーキテクチャ中心のモデルベースシステムエンジニアリング（ACMBSE）を通じて、この目標に向けた最初のステップを踏み出している。

11 参考文献

Autosar Classic 2022, AUTOSAR Classic Platform, release November 2022
<https://www.autosar.org/standards/classic-platform>

Beuche, D, Papajewski, P, Schröder-Preikschat, W. Variability management with feature models, Science of Computer Programming, Volume 53, Issue 3, 2004, Pages 333-352.

Brooks M. D., Wheeler T.M. 2007. Experiences in Applying Architecture-Centric Model-Based System Engineering to Large-Scale, Distributed, Real-Time Systems
<https://www.mitre.org/sites/default/files/pdf/07_0838.pdf > Pub. 2007 Computer Science

Burkacky, O, Kellner, M, Deichmann, J, Keuntje, P and Werra, J. 2021, Rewiring car electronics and software architecture for the 'Roaring 2020s'. McKinsey & Co

Douglas, B, P 2017. Harmony a MBSE Deskbook Version 1.00 Agile Model-Based Systems Engineering Best Practices with IBM Rhapsody. IBM Corporation

Ducamp, C, Bouffaron, F, Ernadote, D, Wirtz, J and Darbin, A. 2022. MBSE approach for complex industrial organization program. INCOSE International Symposium, pp. 839-856.

Forlingieri, M 2022. 'The four dimensions of Variability and their impact on MBPLE: How to approach variability in the development of aircraft product lines at Airbus'. Proceedings of the 16th International Working Conference on Variability Modeling of Software-Intensive Systems (VAMOS '22), February 23–25, 2022, Florence, Italy. ACM, Vamos.

Forlingieri, M. Weilkiens, T. 2022. 'Two Variant Modeling Methods for MBPLE at Airbus'. INCOSE International Symposium, vol. 32, no. 1, pp. 1097-1113.

Friedenthal, S. Moore, A, Steiner, R. 2015. A Practical Guide to SysML: The Systems Modeling Language. The MK/OMG Press, 3rd Edition

IBM 2022, IBM Engineering Systems Design Rhapsody, viewed 13 December 2022
<https://www.ibm.com/products/systems-design-rhapsody>

IBM Docu 8.3, Integrating Rational Rhapsody and the MathWorks Simulink
< https://www.ibm.com/docs/en/elms/esdr/8.3?topic=tools-integrating-rational-rhapsody-mathworks-simulink>

ISO 26262-3:2018. Road vehicles — Functional safety — Part 3:
Concept phase, release 2018

ISO/IEC 26550:2015. Software and systems engineering — Reference model for product line engineering and management.

ISO/IEC 26580:2021. Software and systems engineering — Methods and tools for the feature-based approach to software and systems product line engineering.

ISO/IEC/IEEE 15288:2015, Systems and software engineering — System life cycle processes

ISO/IEC/IEEE 42010:2022. Software, systems and enterprise — Architecture description

Nayak, J, Mishra M, Naik, B, Swapnarekha, H, Cengiz, K, Shanmuganathan, V. 2022. An impact study of COVID-19 on six different industries: Automobile, energy and power, agriculture, education, travel and tourism and consumer electronics. Expert Systems, vol. 39, no.3

OMG 2018, OMG Systems Modeling Language (OMG SysML), Version 1.6.

Seidel, E. Forlingieri, M. 2023, Moving towards Server-Zone Architecture with MBSE at Continental, INCOSE International Symposium.

TOGAF 10, The TOGAF Standard, 10th Edition, Version C220
<https://publications.opengroup.org/standards/togaf/specifications/c220>

UDPM 2017, UPDM Unified Profile for DoDAF and MODAF, Version 2.1.1
<https://www.omg.org/spec/UPDM/2.1.1/PDF>

Weilkiens, T. 2008, Systems Engineering with SysML/UML: Modeling, Analysis, Design, The MK/OMG Press

Weilkiens, T. (2020). SYSMOD - The Systems Modeling Toolbox: Pragmatic MBSE with SysML. MBSE 4U, 3rd Edition

Zachman 2023, The Zachman Framework, Version 16.1
<https://sparxsystems.com/resources/user-guides/16.1/model-domains/frameworks/zachman.pdf>

12 経歴

エンリコ・サイデルは、invenio Systems Engineering GmbH のシニアコンサルタントであり、現在はモデルベース開発の専門家として責任を負っています。テクニカルアーキテクトとして、iSEF の開発をリードしています。以前は、Continental Automotive Singapore のシニアテクニカルエキスパートとしてシステムエンジニアリング（SE）を担当していました。ビジネスエリア（BA）アーキテクチャとネットワーキング内では、エンジニアリングエクセレンスグループにおける SE 作業原則の定義を担当していました。システムエンジニアリングの経験が17年以上あり、チームと共にキャパビリティアーキテクチャフレームワーク（CAF）を共同開発し、現在はその後継製品である iSEF を用いて MBSE のアイデアを推進しています。