

開発に貢献するシステム記述(システムモデル) のコツ ～“お絵かき” SysMLからの脱却～

イノベーティブ・デザインLLC

CEO 石橋金徳

Abstract

システムモデル、システム記述、SysMLなどシステムズエンジニアリング、Model-Basedシステムズエンジニアリング(MBSE)に関連する言葉は日本国内においても自動車関連の開発を中心に非常によく聞かれる様になりました。しかしながら「開発に貢献するシステム記述(システムモデル)」のイメージが持てていない個人や組織が多いのが事実です。本講演ではシステム記述言語SysMLを例に、単なる“お絵かき”から脱却し、システム開発に実際に役に立つシステム記述を実現するための基礎となるコツをご紹介します。

40分

自己紹介

石橋 金徳

kane.ishibashi@innovative-design.jp

イノベティブ・デザイン LLC

CEO/共同創業者、システムズエンジニア

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

特任助教

システムズエンジニアリング、MBSE

大規模複雑システムの開発 企業共同研究 推進

自動車産業におけるモデル利用のあり方に関する研究会(モデル流通) 2014/2015 有識者 (経産省)
第四次産業革命スキル習得講座(自動車分野のモデルベース開発) 2017~2020 審査委員 (経産省)

<経歴>

東京大学 超小型衛星戦略研究センター

超小型人工衛星開発

開発プロジェクトマネージャ/システムズエンジニア

株式会社本田技術研究所 二輪R&Dセンター

二輪車エンジン設計

電動パーソナルモビリティ研究開発

米国University of Minnesota, Bachelor of Mechanical Engineering

慶應義塾大学大学院 システムズエンジニアリング修士

2020/10/20



innovative DESIGN

イノベーター・デザインLLC

デザインファームとして、システムズエンジニアリング、Model-Basedシステムズエンジニアリングを応用することで、プロダクト、サービス、システム、企業の戦略、経営など様々な領域における”イノベーターにデザインし実現すること”を支援

Design Consulting

- ビジネス戦略策定、大規模開発戦略、システムデザイン、システム開発などの分野で、クライアントのチームとともに課題を解決

Research

- 共同研究、委託研究、などの形態で調査や研究を実施

Education

- クライアントの個人やチームに対してシステムズエンジニアリング、MBSEなどの教育を提供

Reseller

- システムモデリングツールであるNo Magic, Inc.製MBSE関連製品(ソフトウェア)の日本国内への販売

導入

システムズエンジニアリング、MBSEとは？

- システムズエンジニアリングとは大規模・複雑なシステムを実現するための体系化された考え方であり、もともと米国の航空宇宙や防衛の開発などにおいて体系化された背景を持つ。
- システムの目的、背景を明確化し、ライフサイクル、コンテキストなどの分析的な理解進め、システムの機能設計(ふるまい)・物理設計(実現手段)すなわちアーキテクチャの設計を行う。また、システムのV&V(検証と妥当性確認)についても戦略性を持って計画、実施を行う。
- 複数の専門分野を統合的に扱うアプローチである。これを実践するシステムズエンジニアは開発全体を牽引するキーパーソンとなる。
- MBSEとは、このようなシステムズエンジニアリングを様々な種類のモデルを用いることで電算機の力を最大限に活用しながら、非常に大規模で複雑な対象物に対しても効果的、効率的に実施するための手段である。
- とくに、セマンティック技術を応用したシステム開発の効率化、高度化に注目が集まっている。

なぜシステムズエンジニアか？

- 求められるより**高度な設計説明責任**
 - 機能安全、ISO26262
 - 米国Department of Transportation, NHTSA “Automated Vehicles for Safety”
<https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety>
- 指数関数的に増加する**設計の複雑さ、検証の難しさ**
 - ADAS、自動運転、自動車のネットワーク化、各種機能部品の電動化
 - ハイパーコネクティビティ、次世代移動通信システム
 - クラウド化、Everything as a Service、XaaS
- 必須になる**高度なシステム分析と判断能力**
 - システム全体を見渡したFTA(演繹的リスク分析)
 - システムのフェールを検知し、システムをフェールから守り、リカバリーする方を柔軟に検討し設計する
- 求められるエンジニアリングにおける**高い戦略性**
 - 設計、検証の効果的な外出しによる開発工数削減
 - モジュール化、共通アーキテクチャ、インタフェース標準による開発コスト削減
 - ローカル/グローバルの自在な調達、ローカル/グローバルな事業展開
- “モノづくり”の**視野の拡大**
 - IoT
 - Industry 4.0、スマートファクトリー
 - Reference Architecture Model Industry 4.0 (RAMI4.0) <http://www.omg.org/news/meetings/tc/berlin-15/special-events/mfg-presentations/adolphs.pdf>
 - Smart Grid Architecture Model (SGAM) Framework
http://ec.europa.eu/energy/sites/ener/files/documents/xpert_group1_reference_architecture.pdf

なぜModel-Basedシステムズエンジニアリングか？

- 複雑さを扱い易くする**段階的詳細化**
 - システムを表現する粒度を複数階層で制御し、俯瞰と詳細が常に同期された状態を実現する
- 開発に関わる**膨大な情報から高度な判断が必要**
 - 数値データだけでなく、目的、制約、意図等も併せて扱う
- 求められる高い**変更分析**および**変更管理能力**
 - 複数領域にまたがる変更インパクト分析、全体俯瞰的な変更管理
- 自由度の高いパートナーシップや調達への迅速な対応
 - 意思決定の伝達、情報のやり取り、などの管理とスピードをバランスする
- 開発環境もこうした変化に合わせて急速に進化中
 - CAE・MBDとの相互連携、PLMの進化、OSLCへの期待の増大

システムズエンジニアリング、MBSEのモチベーション構造

Better productのために
Better engineeringを実現したい!

- ・トレーサビリティをしっかり確保
- ・変更インパクトを即座に把握
- ・効果的かつ確実な検証
- ・効率的な開発プロセスなど

そのために

Systems Engineering

システムズエンジニアリングのアプローチを
効果的、効率的にとりたい (More efficient and more effective SE)

そのために

Model-based Systems Engineering

モデルベースで行うシステムズエンジニアリングを
国際的に認められた共通言語で行いたい (With “common” language)

そのために

SysMLによるMBSE

決してSysMLやMBSEがモチベーションの
源泉や目的になってはならない

開発に貢献するシステム記述 (システムモデル)のコツ

- ① 記号を並べる、ではなく、情報を構造化する
- ② Modeling(モデリング)に注力、ではなく、
Model Consumption(モデル活用)に注力
- ③ 可視化、ではなく、問題・課題を解決する

開発に貢献するシステム記述 (システムモデル)

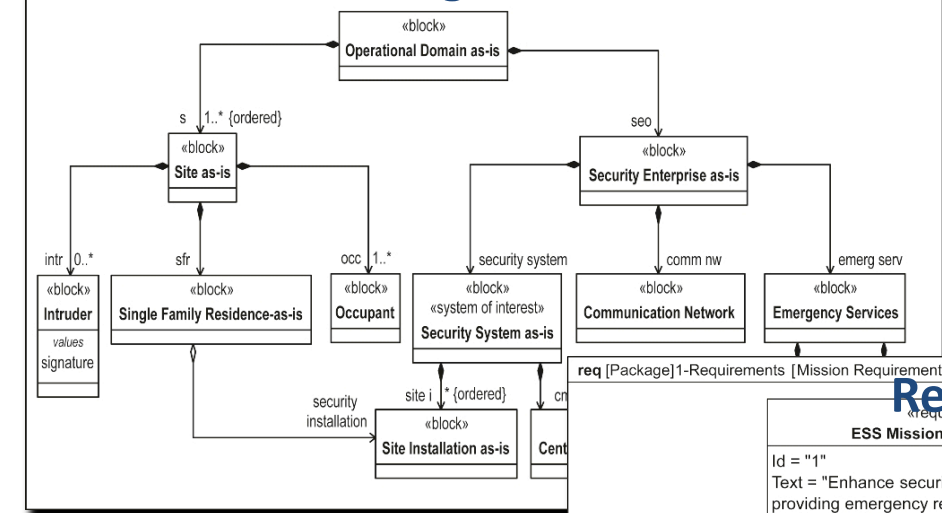
Executive Summary

「開発に貢献するシステム記述(システムモデリング)」 の考え方と記述言語“SysML”の使い方

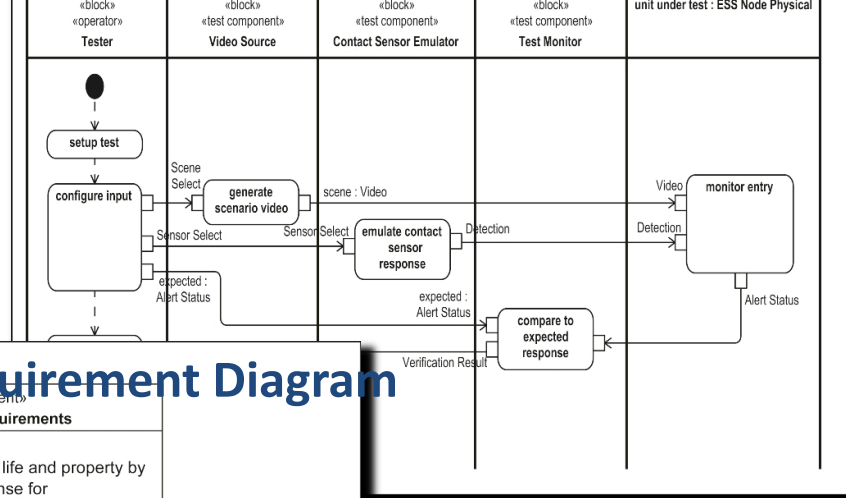
- 「システム記述」とは単純な図解のことではなく、システム開発を進める上で必要になる技術分野横断的な対象全体を俯瞰した各種エンジニアリング情報を、情報要素と情報要素間の関係性を定義し、構造化することである。すなわち、情報構造体の構築である。
また、システム記述言語は、この構造を電算機で処理可能にするための表記法である。
 - MBSEにおいて「System Modeling」(動詞)と称されることもある。
- 情報要素、情報要素間の関係性は、「メタ構造」と「構造」で定義される。「メタ構造」で定義された要素の基礎定義、関係性の基礎定義を用いて、実際のシステムに関する情報の「構造」を構築するのである。
 - メタ構造は“概念の構造”であり「メタモデル」や「オントロジー」とも呼ばれる。
 - 参考資料: Vitech, “One Model, Many Interests, Many Views”, 2017.
 - システム記述言語SysMLにおいては、<<block>> <<allocate>>などのstereotype(要素の定義)とその関係性の定義がメタ構造にあたる。
 - SysMLを用いた実際のシステム記述においては、目的と対象に合わせてカスタムまたはオリジナルのstereotypeとその関係性の定義を複数作成し、それを用いてシステム記述を行い、システムに関する情報の構造を構築する。
- システムに関する情報の構造体は、メタ構造を辿る事によって、任意の概念のつながりを辿りながら目的に合わせた情報群を抽出することが可能である。これを情報構造体の活用と言う。
 - SysMLのオーサリングツールの1つであるCATIA/No MagicのCameo Systems Modelerにおいては「Structured Expression」と呼ばれる機能で、メタ構造を辿る情報の指定や抽出が可能である。
 - 中でも、「Metachain Navigation」と呼ばれる辿り方を用いることで、連続的にメタ構造をたどっていくことが可能であり、非常に自由度の高い指定や抽出が可能である。
- システム開発の問題や課題の解決の手段の1つとして、情報構造体を構築し、活用するのである。13

よくあるMBSEの誤解：システムモデル＝SysML(のDiagram)

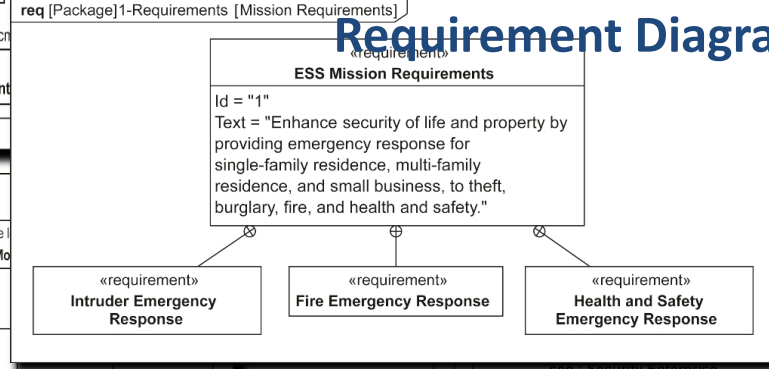
Block Definition Diagram



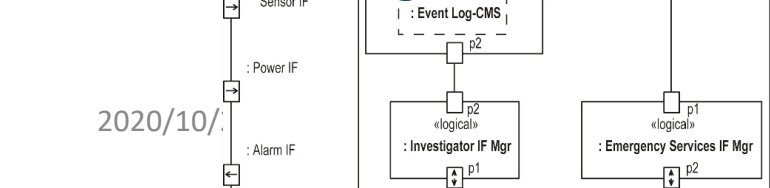
Activity Diagram



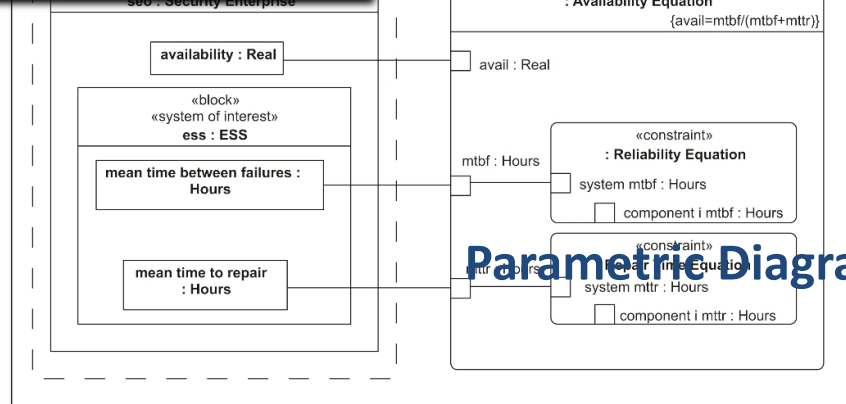
Requirement Diagram



Internal Block Diagram

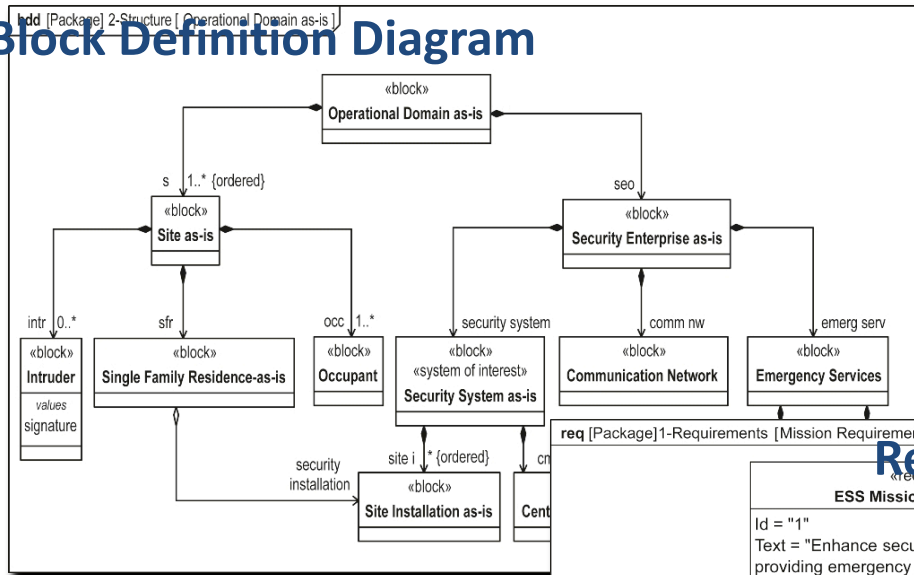


Parametric Diagram

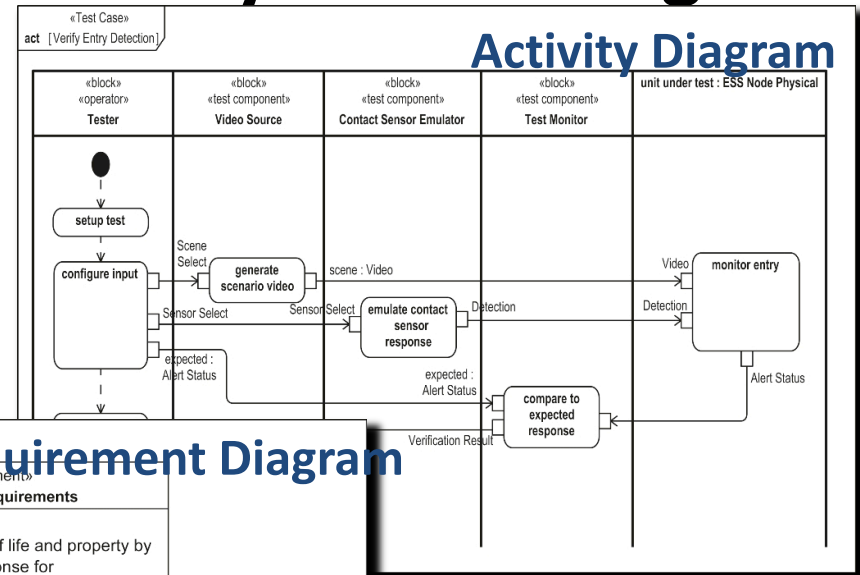


よくあるMBSEの誤解：システムモデル＝ SysML(のDiagram)

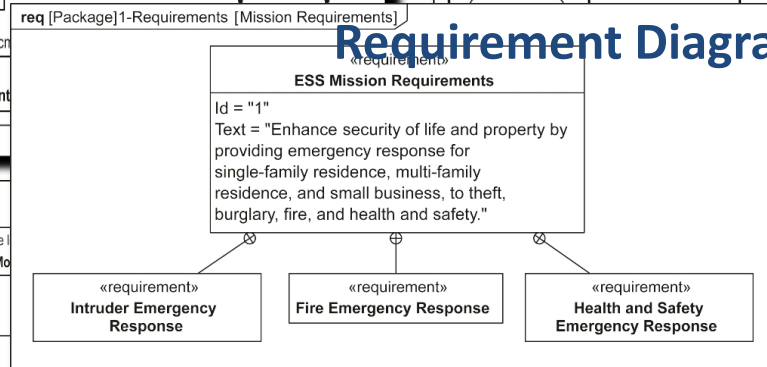
Block Definition Diagram



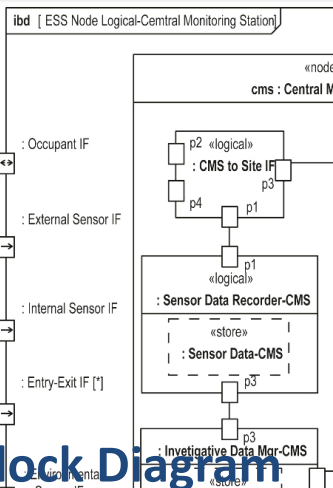
Activity Diagram



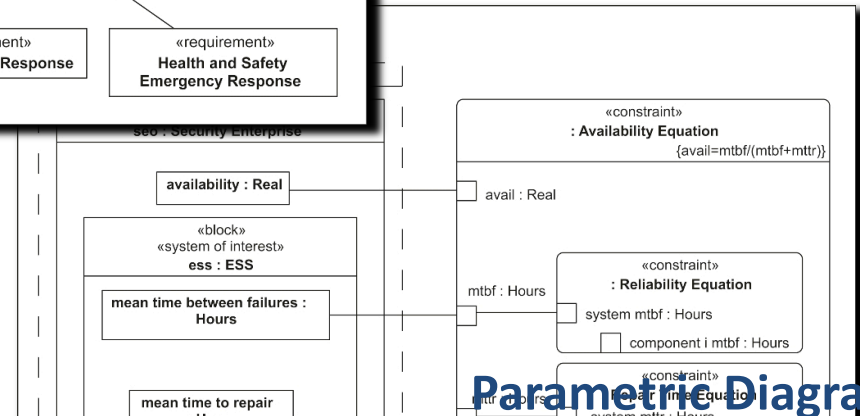
Requirement Diagram



Internal Block Diagram



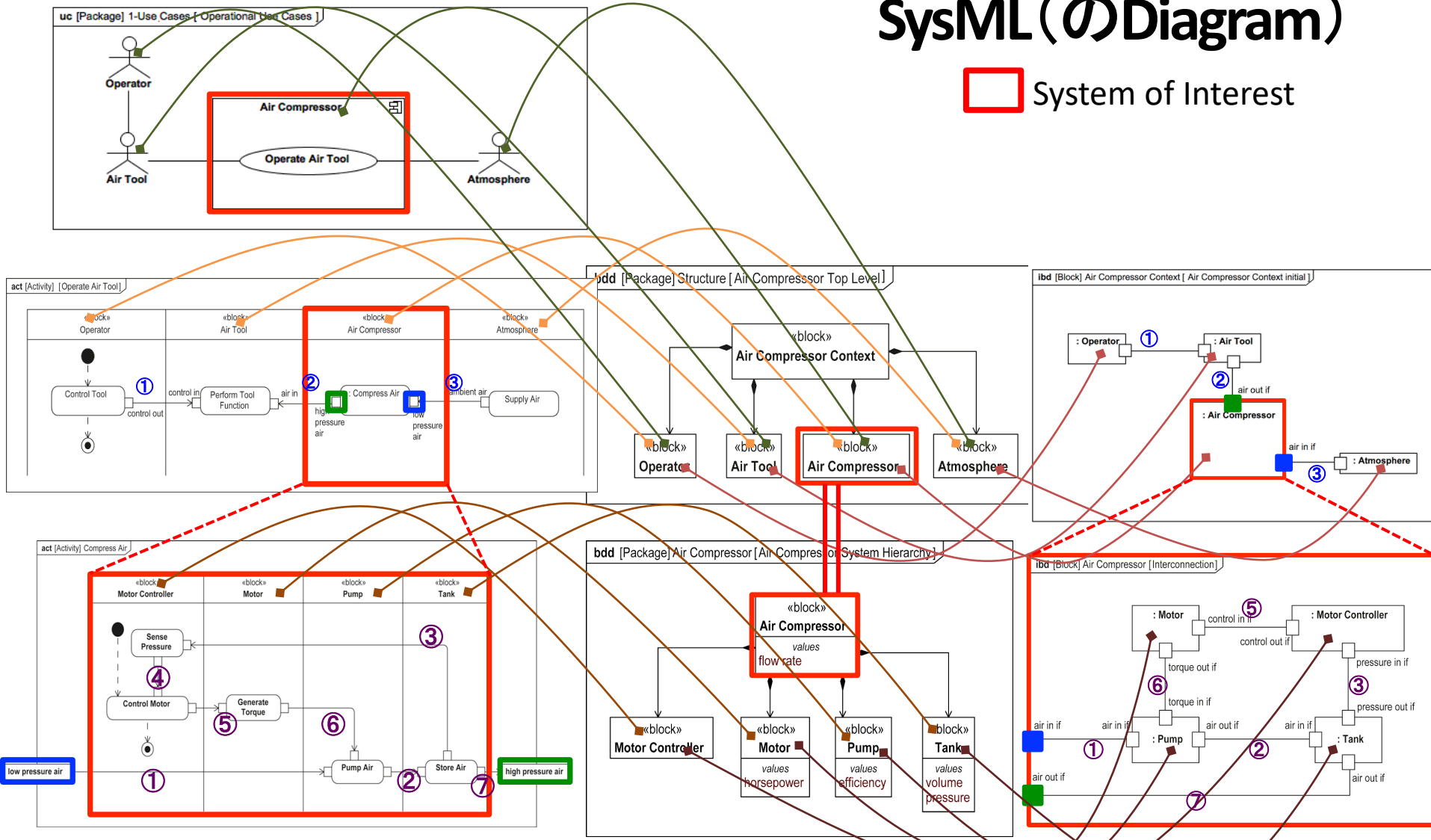
Parametric Diagram



「図解して理解が進む」「図によってコミュニケーションし易い」は
SysMLのDiagramの一側面に過ぎない。

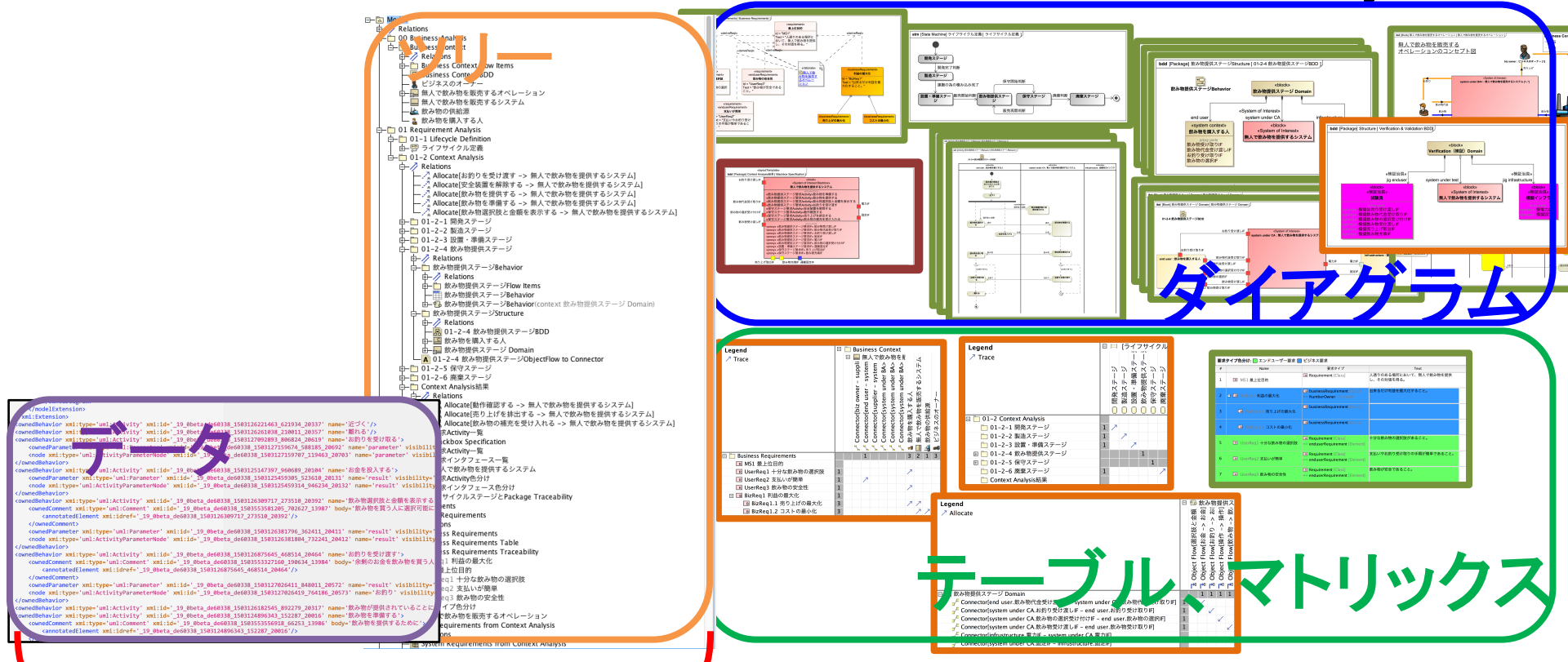
MBSEにおける「システム記述」としての SysML(のDiagram)

 System of Interest



システムに関する様々な情報が構造的、階層的に整理され、一貫性を保って記述されることで、システムの「全体俯瞰」の一助となる。

MBSEにおける「システム記述」としてのSysML

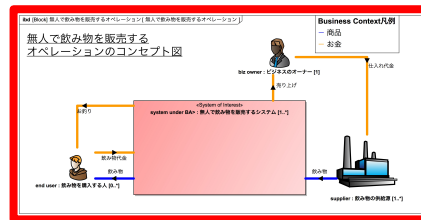
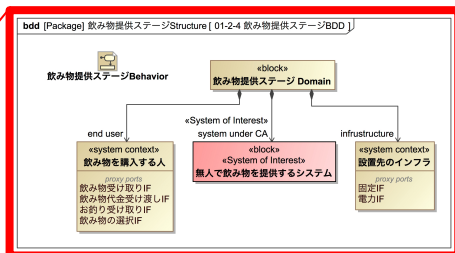
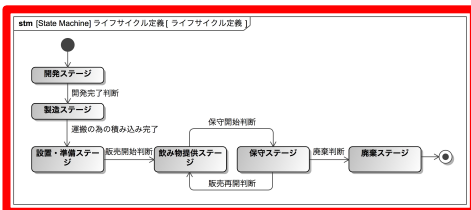


システムモデル

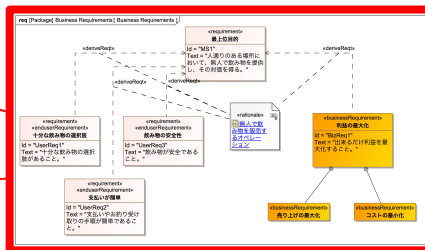
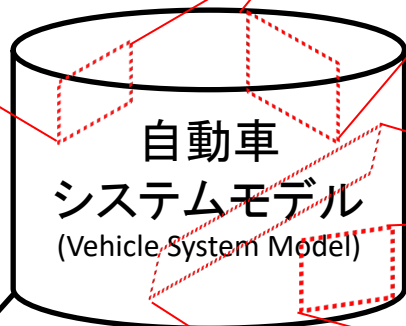
ダイアグラム、テーブル、マトリックスなどはいずれもシステムモデルを意図に合わせて切り取ったRepresentation(表現、見せ方)である。

MBSEにおける「システム記述」の正しい捉え方

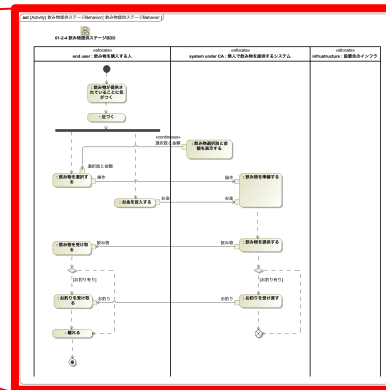
活用用途に合わせて、**任意の情報のみを含んだ任意の形態の断面図**を抽出することが可能な**情報構造体**



例:



ABC Subsystem	Apple Unit
	Purple Device
	Texas Unit
DEF Subsystem	Peach Unit
	Black Device
	Ohio Unit
GHI Subsystem	Cherry Unit
	Blue Device
	Idaho Unit



Item	Specs	Supplier	Weight
Engine	Briggs and Stratton 15 hp Model # 28N787	Central Equipment	13.0
Peerless Transaxle	Item # 13-1476, 3 forward speed, 1 reverse	Surplus Center	14.6
Hubs	1500 lb Spindle Super Lube		22.4
Tires	4.00/15SL Tru-Trac Front Tire		5.6
Rims	3x15	Bevins Motor Co.	3.4
Tubes	5.00-15/4.00-15	S&S Tire	3.6
Rear wheel	26x12.00-12NHS	From surplus	6.7
3 point Hitch Lift	Cat. 0 electric lift	Koplan (Amazon)	7.5
Seat	Deluxe midback steel	Northern Tool & Equipment	9.0
Steering wheel	Replacement, Massey Ferguson keyed hub	Northern Tool & Equipment	11.5
Pillow Block Bearings	1-1/4 UPC206-20	Bearings.com	11.5

システム記述言語(System Description Language)を用いて、技術分野横断的に、対象全体を俯瞰した各種エンジニアリング情報が関係性を持って一元的に整理された、**情報の構造体**

#	Name	要求タイプ	Text
1	MS1 最上位目的	Requirement (Class)	人通りのある場所において、無人で飲み物を提供し、その対象を得る。
2	businessRequirement (Class)	businessRequirement (Class)	NumberOwner (Class)
3	businessRequirement (Class)	businessRequirement (Class)	businessRequirement (Class)
4	businessRequirement (Class)	businessRequirement (Class)	businessRequirement (Class)
5	businessRequirement (Class)	businessRequirement (Class)	businessRequirement (Class)
6	businessRequirement (Class)	businessRequirement (Class)	businessRequirement (Class)
7	businessRequirement (Class)	businessRequirement (Class)	businessRequirement (Class)

Legend	Trace
01-2 Context Analysis	
01-2-1 開発ステージ	
01-2-2 製造ステージ	
01-2-3 設置・準備ステージ	
01-2-4 飲み物提供ステージ	
01-2-5 保守ステージ	
01-2-6 廃棄ステージ	
Context Analysis結果	

システムズエンジニアリングのModel-Basedなアプローチを議論するための5+1のアジェンダ

これら全部合わせて
MBSE

大規模・複雑なシステムを開発する

Process (考え方)

システムズエンジニアリング
(ISO15288), 各社オリジナル, など

supported by ↓

↑ support

大規模・複雑なシステムを開発する

Methodology (方法論)

システムをモデル化してエンジニアリングを進める方法論
①システムとしてエンジニアリングする方法論
②システムをモデル化(情報構造化)して活用する方法論

ISO42010,
OOSEM(INCOSE),
各社オリジナル, など

supported by ↓

↑ support

大規模・複雑なシステムを開発する

**Language/Metamodel/Model
(記述言語/情報構造体)**

SysML, UPDM, OPM,
IDEFO, 各社オリジナル, など

supported by ↓

↑ support

大規模・複雑なシステムを開発する

Tool (ツール、ソフトウェア)

MagicDraw, Rhapsody,
EA, 各社オリジナル, など

supported by ↓

↑ support

大規模・複雑なシステムを開発する

Environment (IT環境)

ネットワーク環境,
PLM/PDM製品,
各社オリジナル, など

大規模・複雑なシステムを開発する



資質、経験、情熱、
アイデア、マインド、
技術力、ソフトスキル、
組織、役割、など

開発に貢献するシステム記述 (システムモデル)のコツ

- ① 記号を並べる、ではなく、情報を構造化する
- ② Modeling(モデリング)に注力、ではなく、
Model Consumption(モデル活用)に注力
- ③ 可視化、ではなく、問題・課題を解決する

開発に貢献するシステム記述(システムモデル)

コツ①

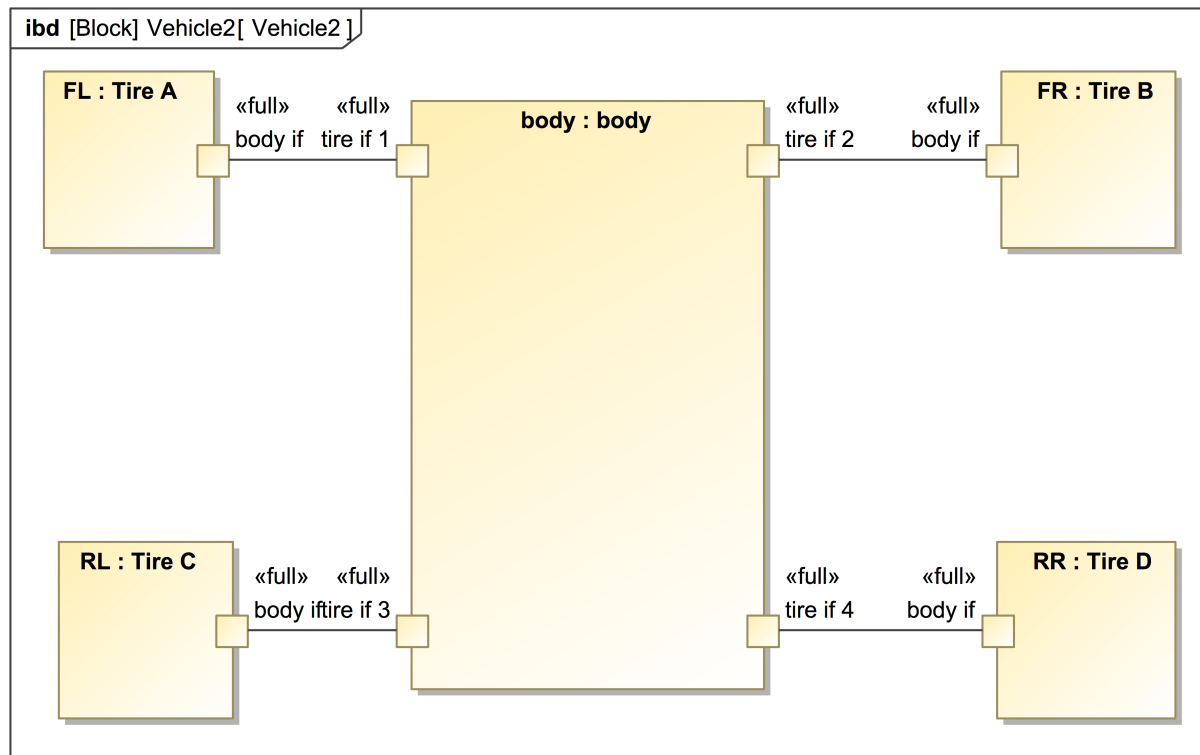
記号を並べる、ではなく、情報を構造化する

コツ①記号を並べる、ではなく、情報を構造化する

- システム開発をうまく進めるために、システムの設計および開発に関わる情報を整理し、構造化する。

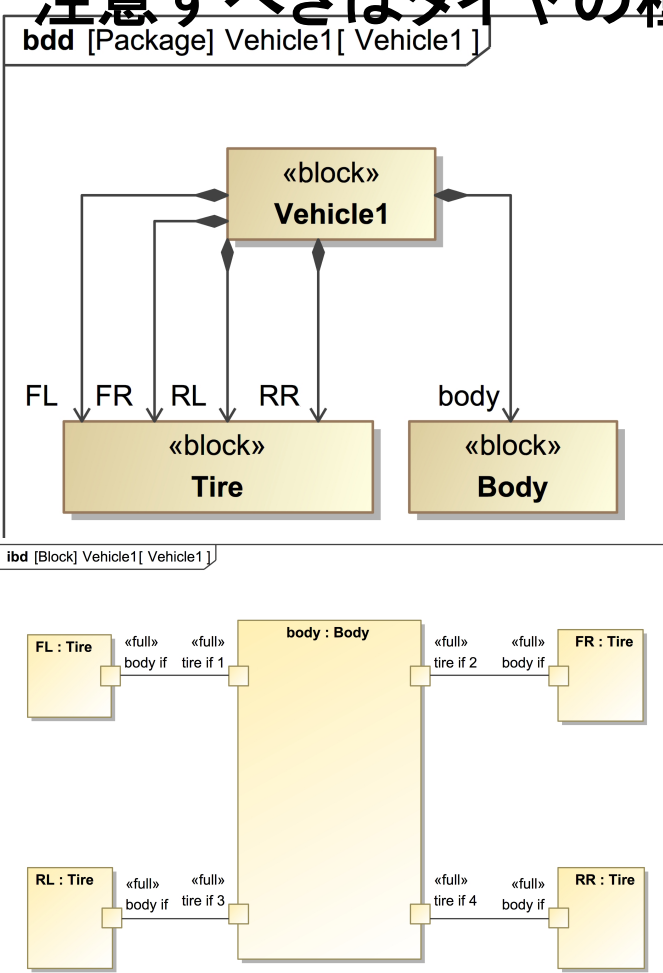
①-1

“お絵かき” SysML例



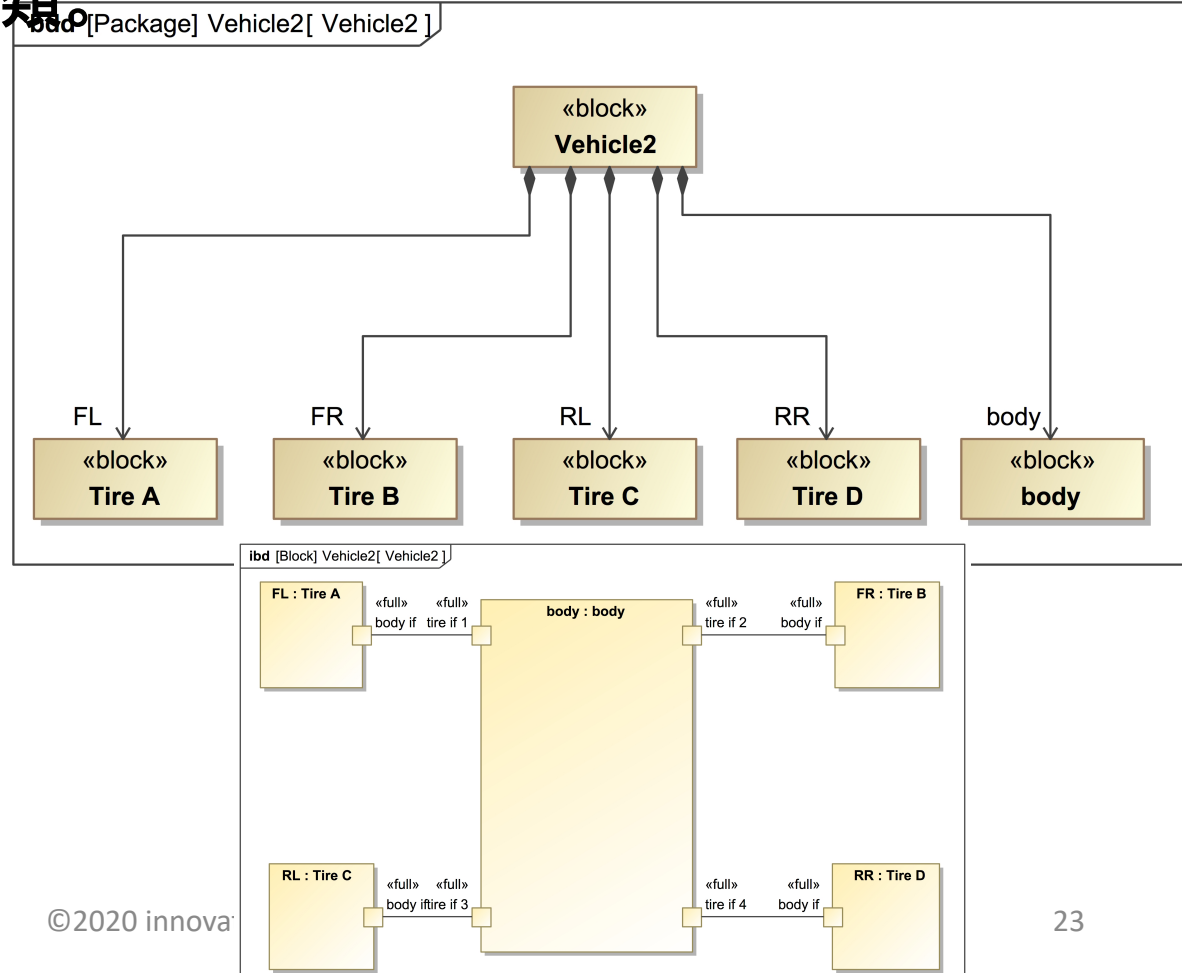
①-1 A情報の構造化例

通常の市販車のタイヤ交換をするサービスエンジニアが作ったシステムモデル。
4つのタイヤは全く同一であり、
注意すべきはタイヤの種類。



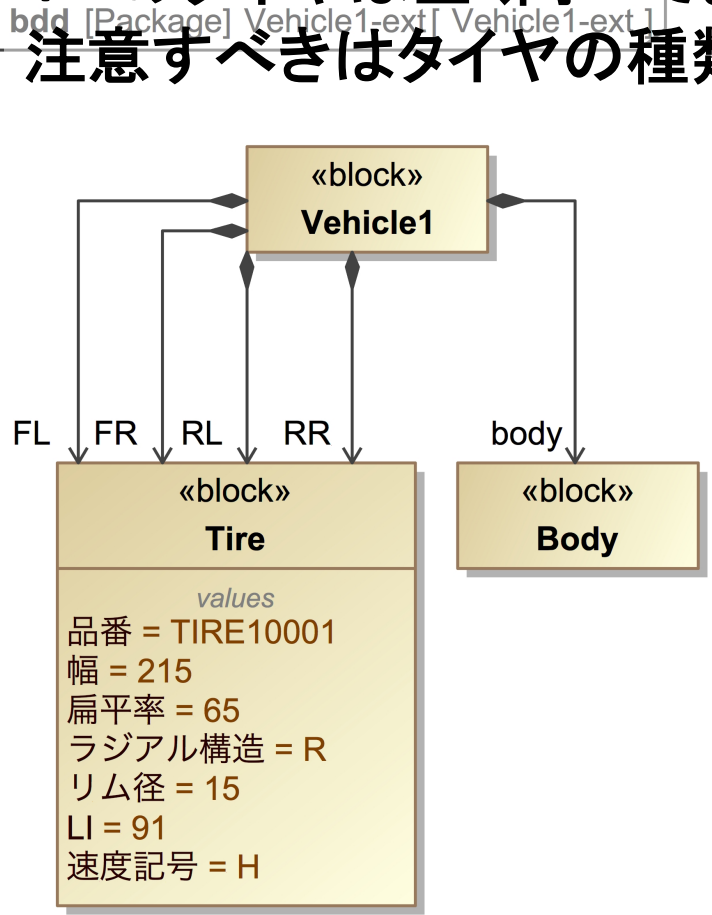
①-1 B情報の構造化例

F1車のタイヤ交換をするピットクルーが作ったシステムモデル。
4つのタイヤは全く別に管理されており、1つずつの特性が重要。



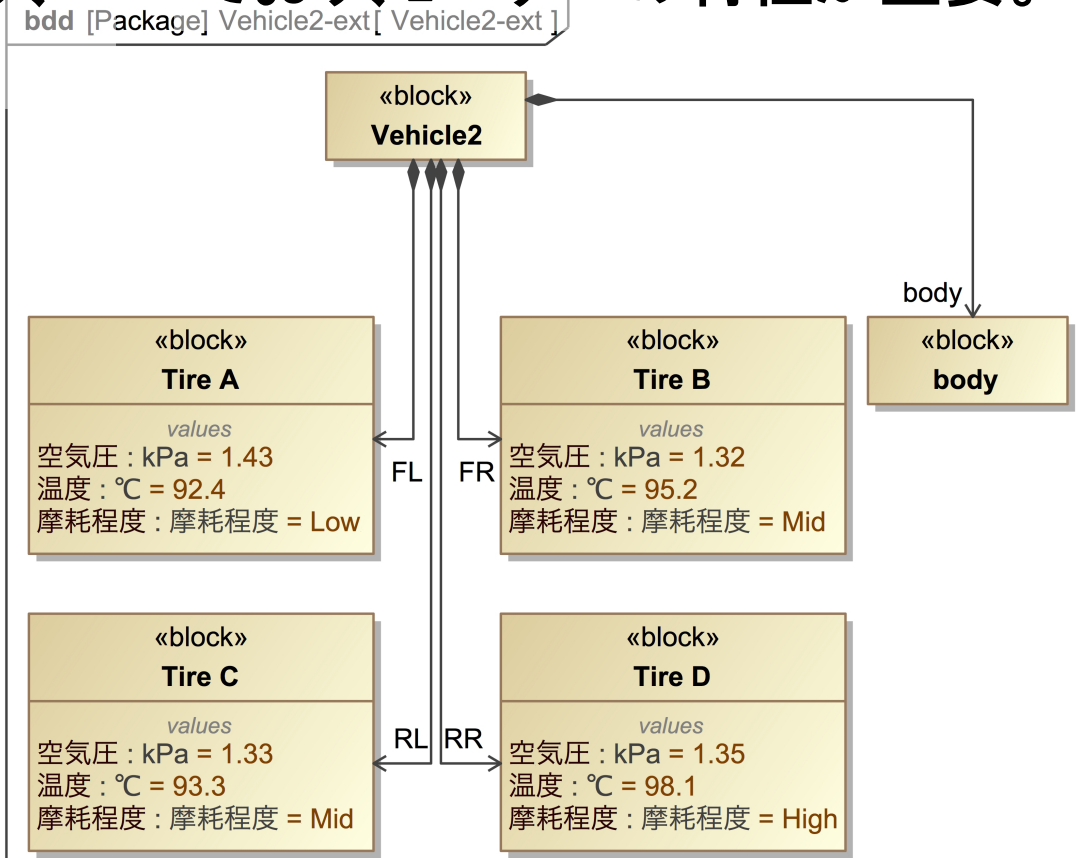
①-1 A情報の構造化例

通常の市販車のタイヤ交換をするサービスエンジニアが作ったシステムモデル。
4つのタイヤは全く同一であり、
注意すべきはタイヤの種類。



①-1 B情報の構造化例

F1車のタイヤ交換をするピットクルーが作ったシステムモデル。
4つのタイヤは全く別に管理されており、
1つずつの特性が重要。

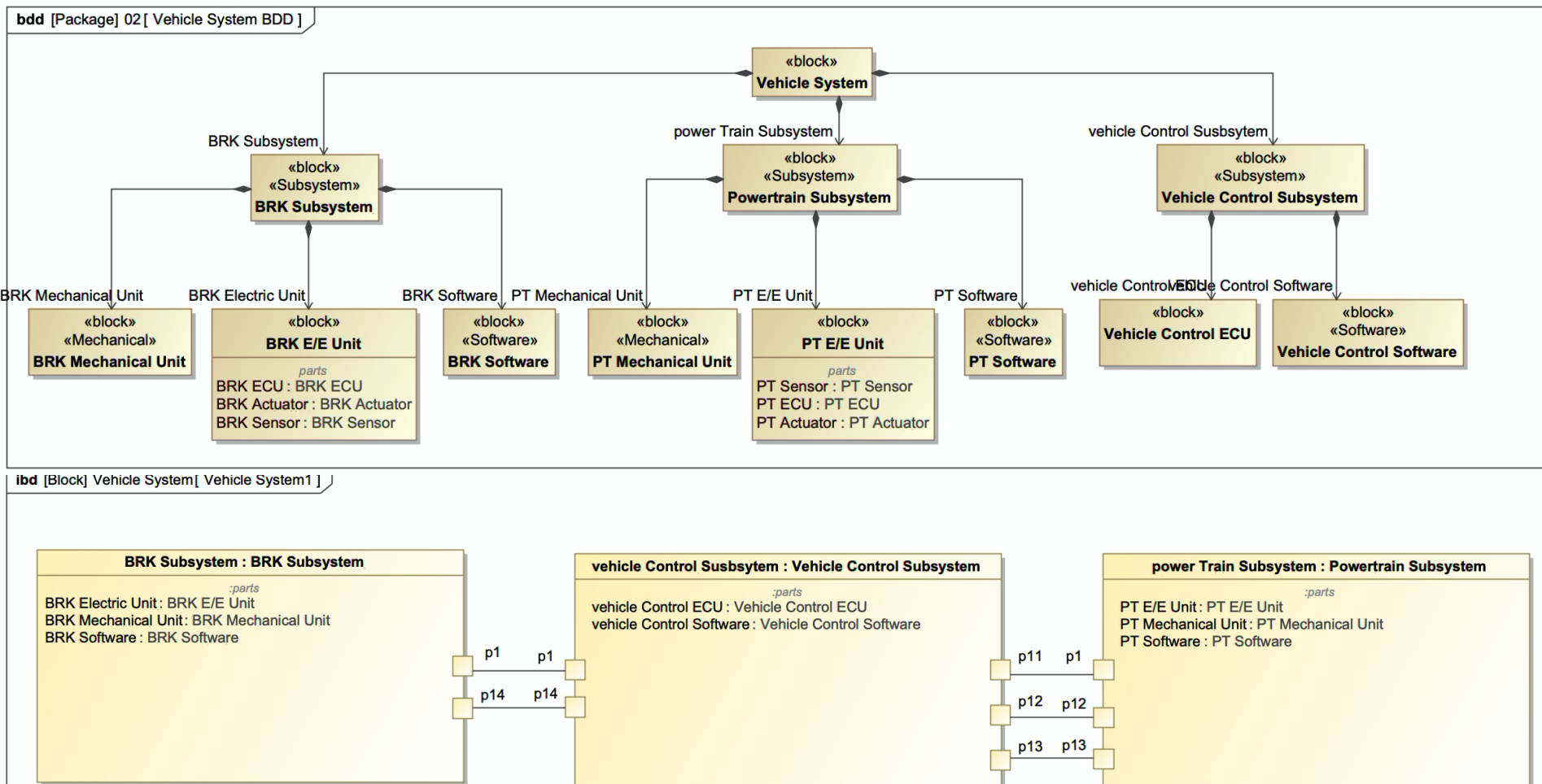


コツ①記号を並べる、ではなく、情報を構造化する

- システム開発をうまく進めるために、システムの設計および開発に関わる情報を整理し、構造化する。

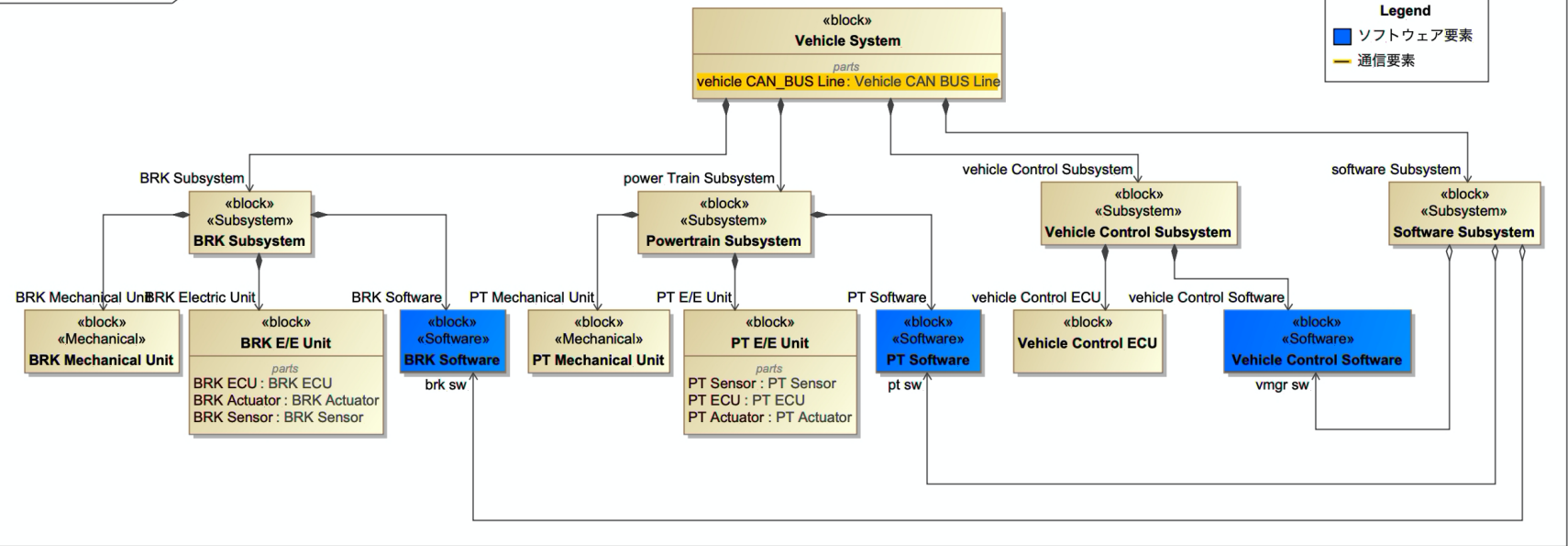
①-2

“お絵かき” SysML例

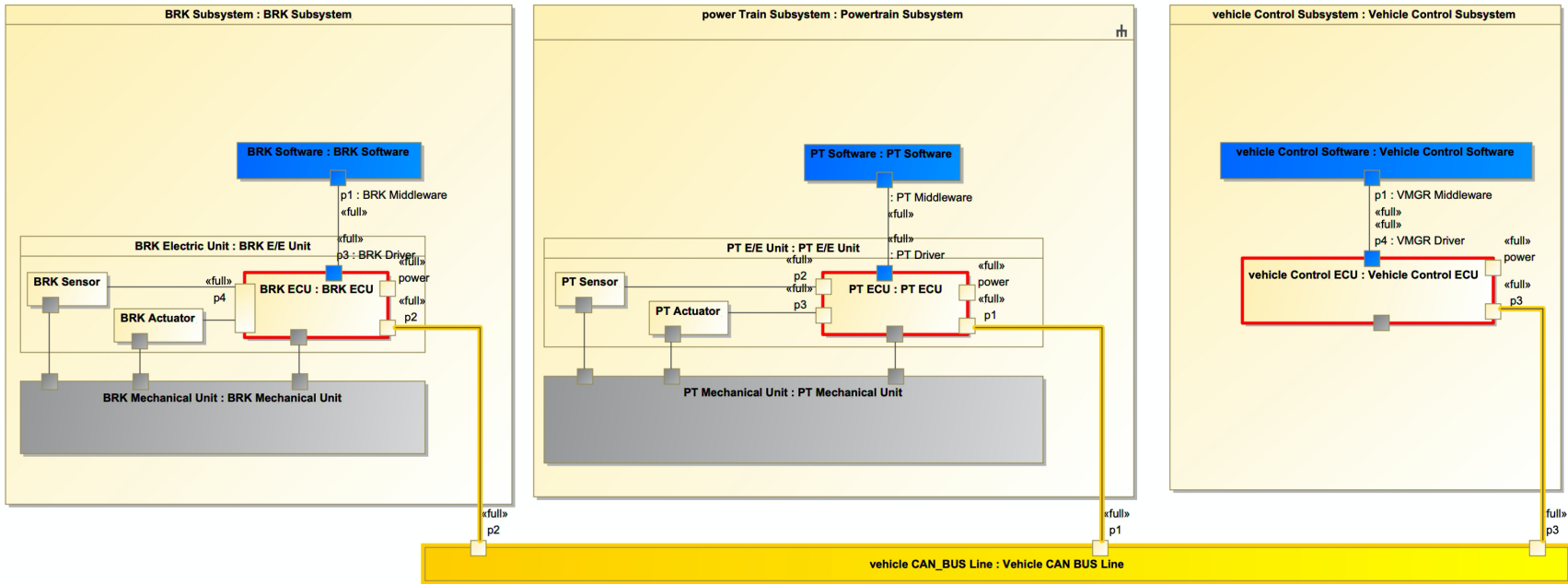


①-2情報の構造化例

bdd [Package] 01 [BDD]

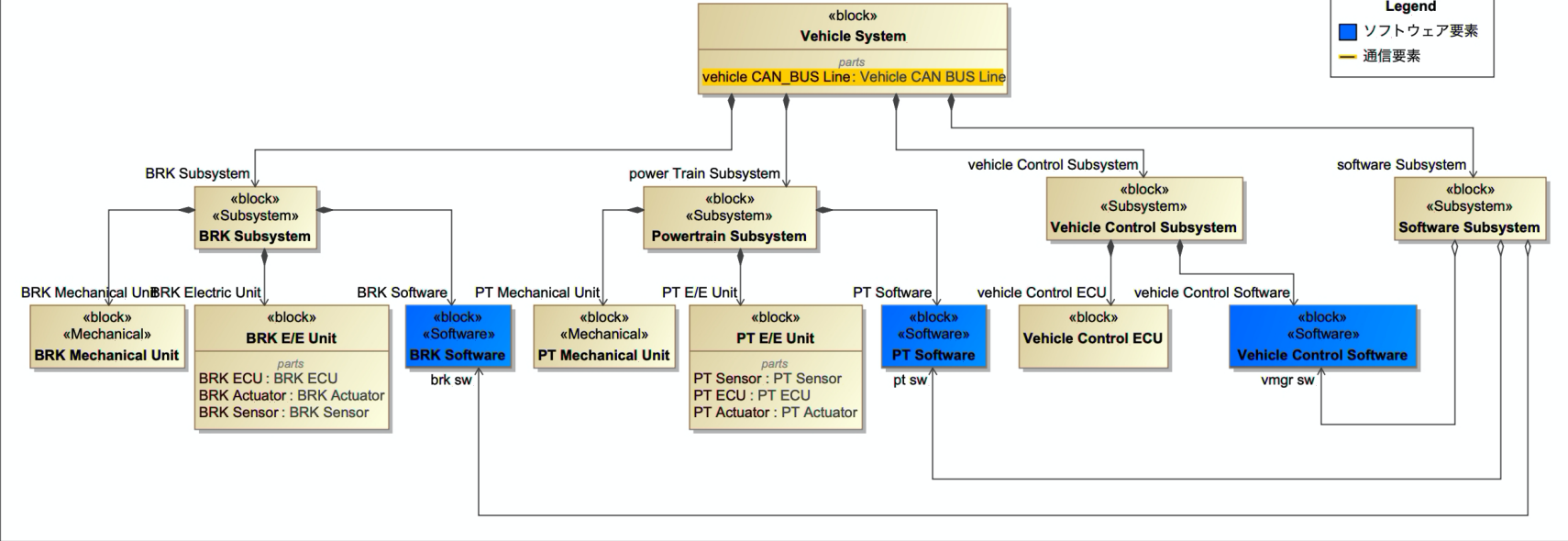


メカ要素 (Mechanical Element)
メカインタフェース (Mechanical Interface)

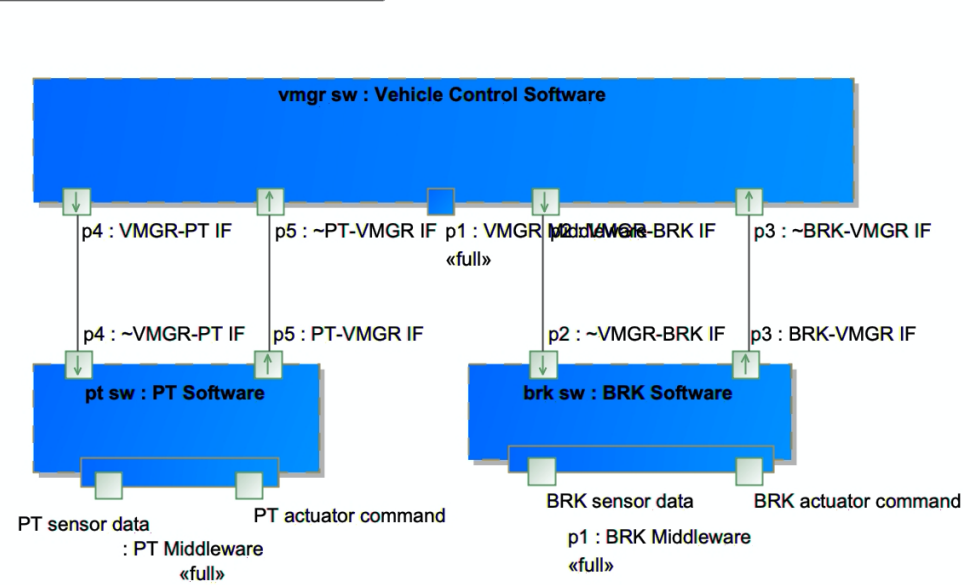


①-2情報の構造化例

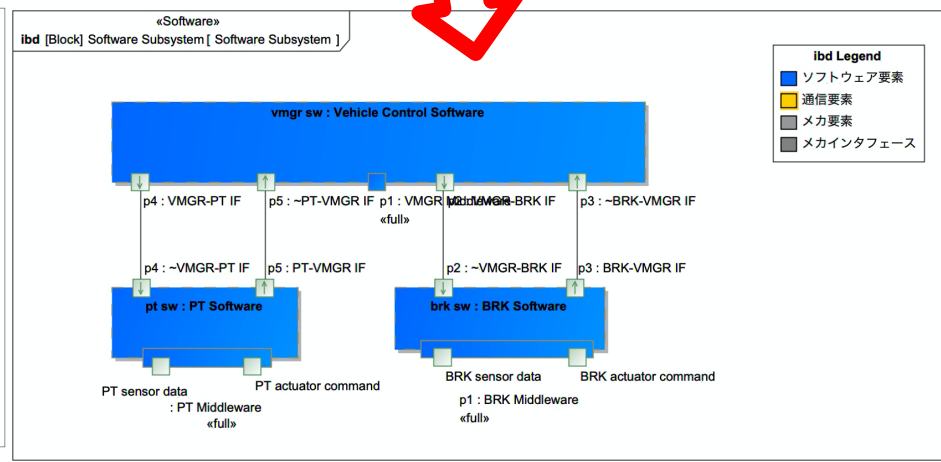
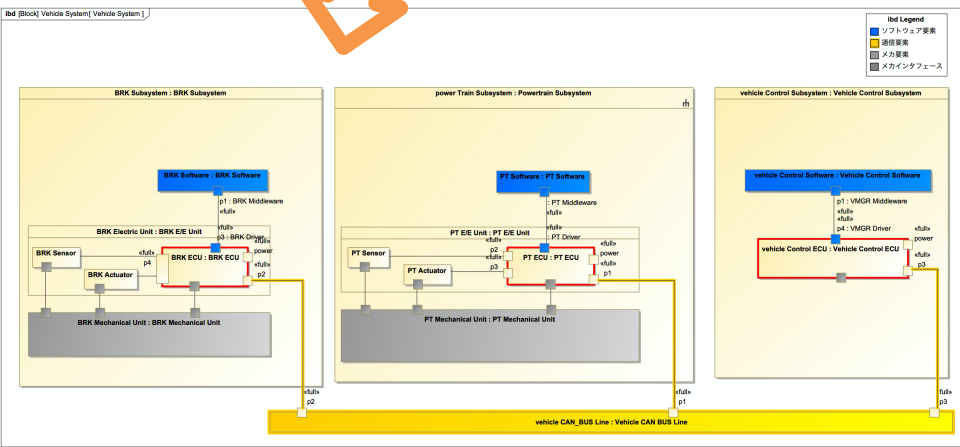
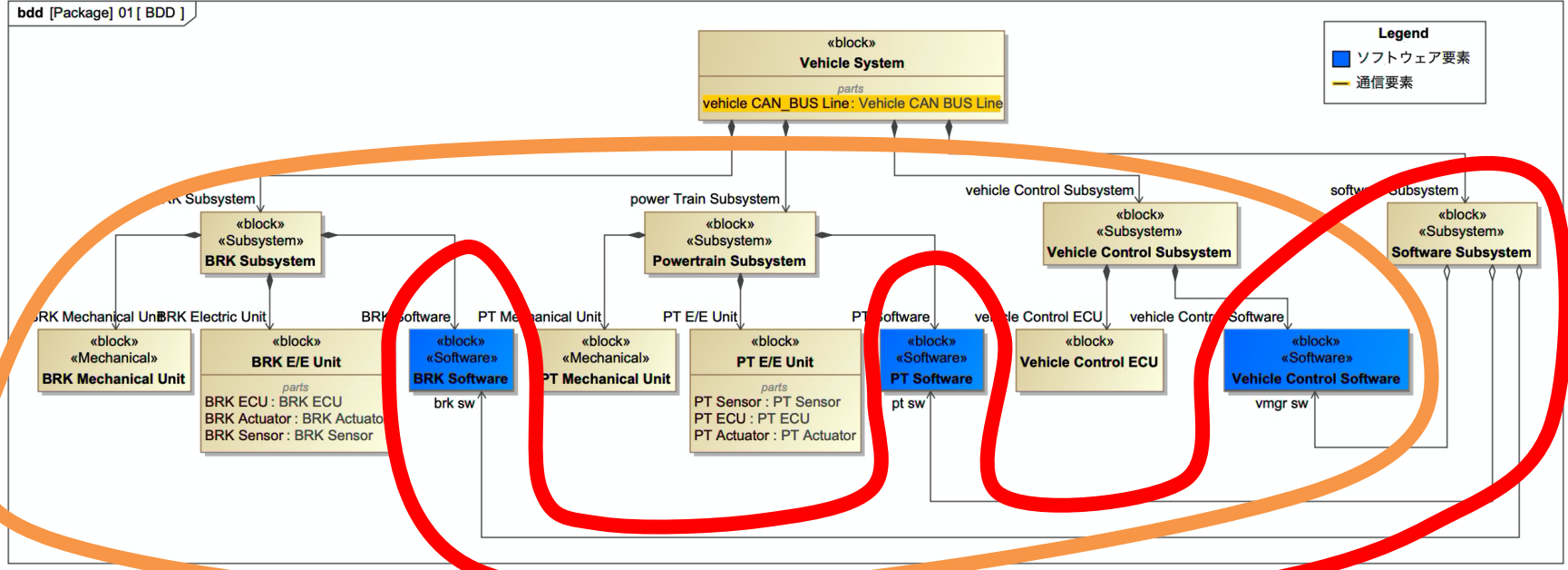
bdd [Package] 01 [BDD]



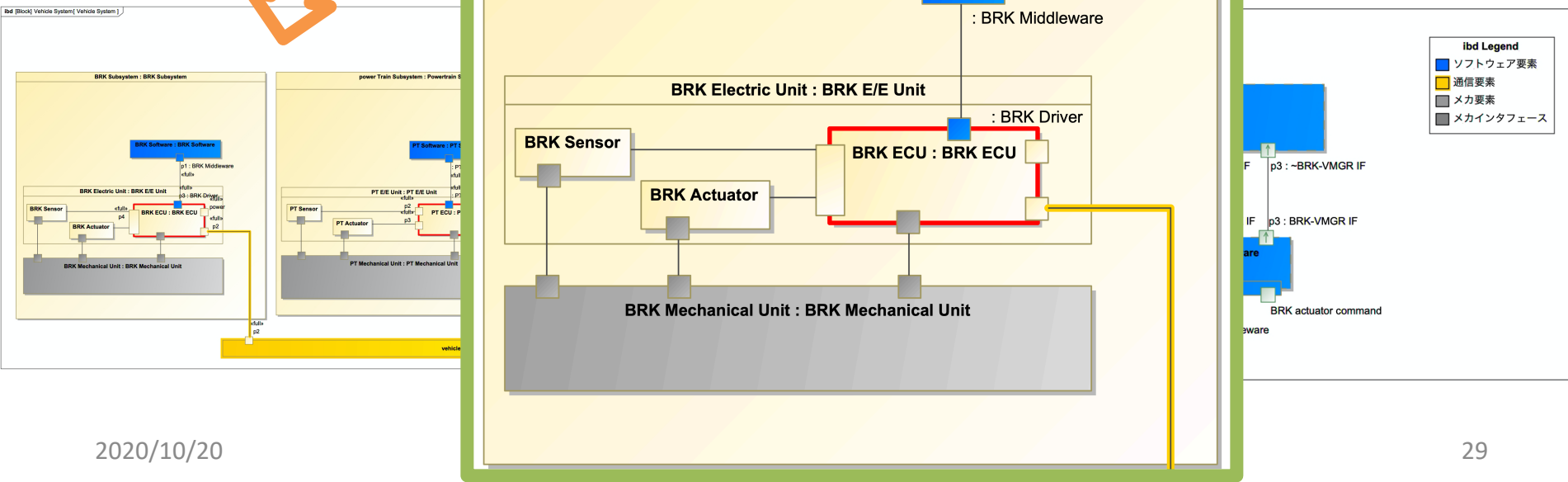
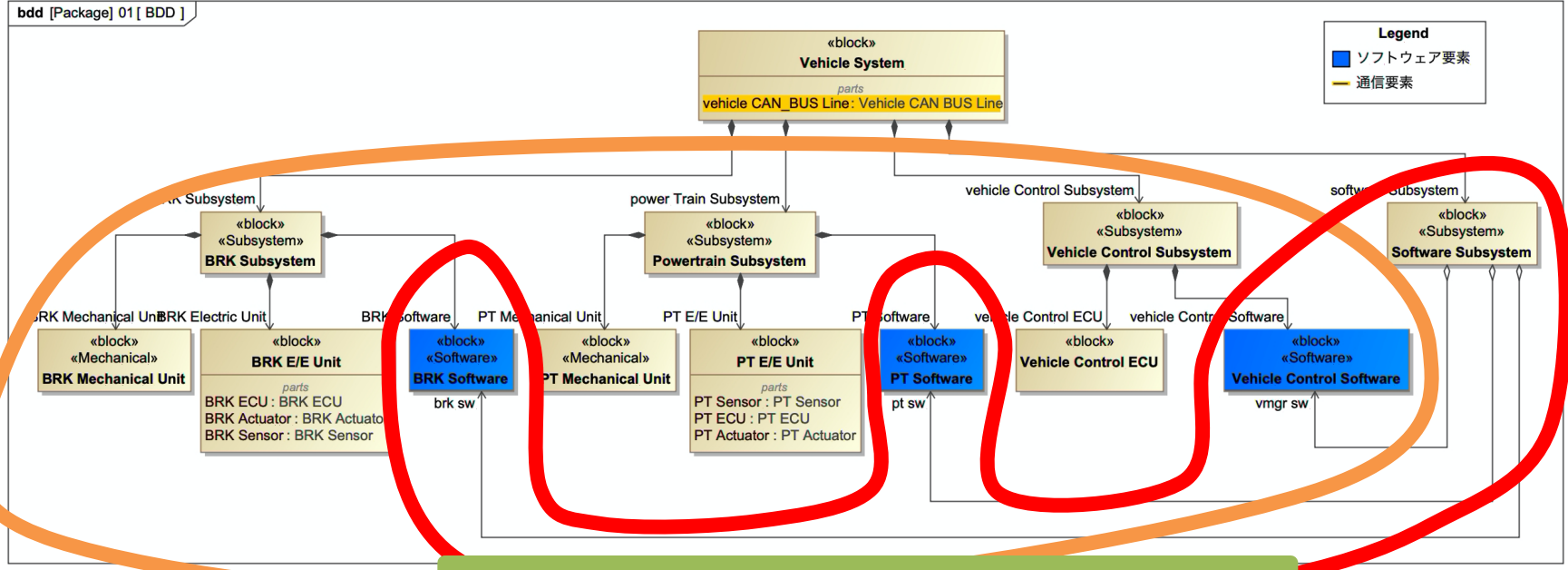
«Software»
ibd [Block] Software Subsystem [Software Subsystem]



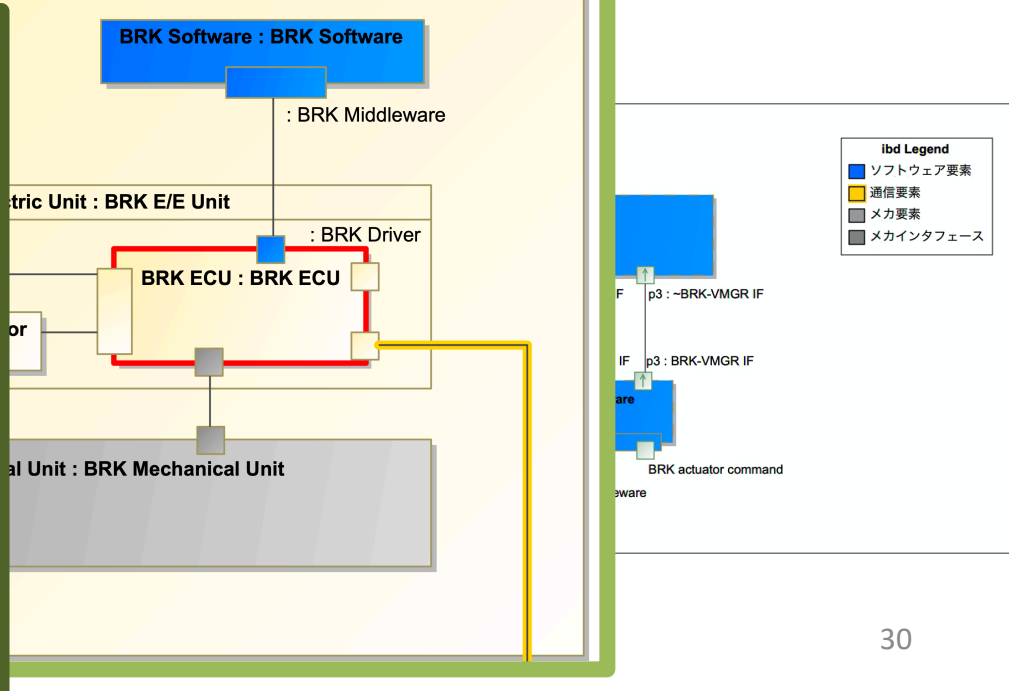
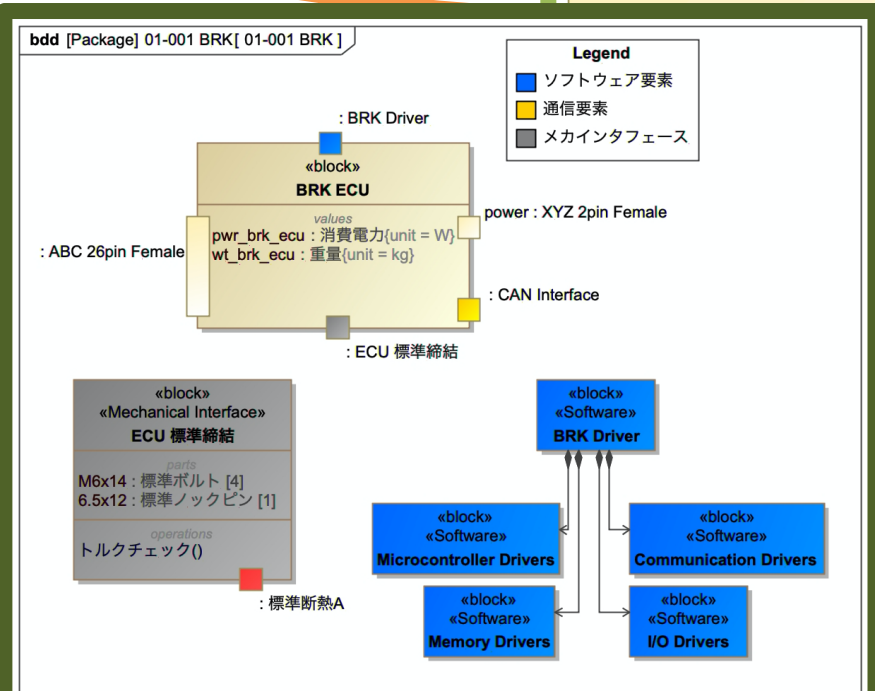
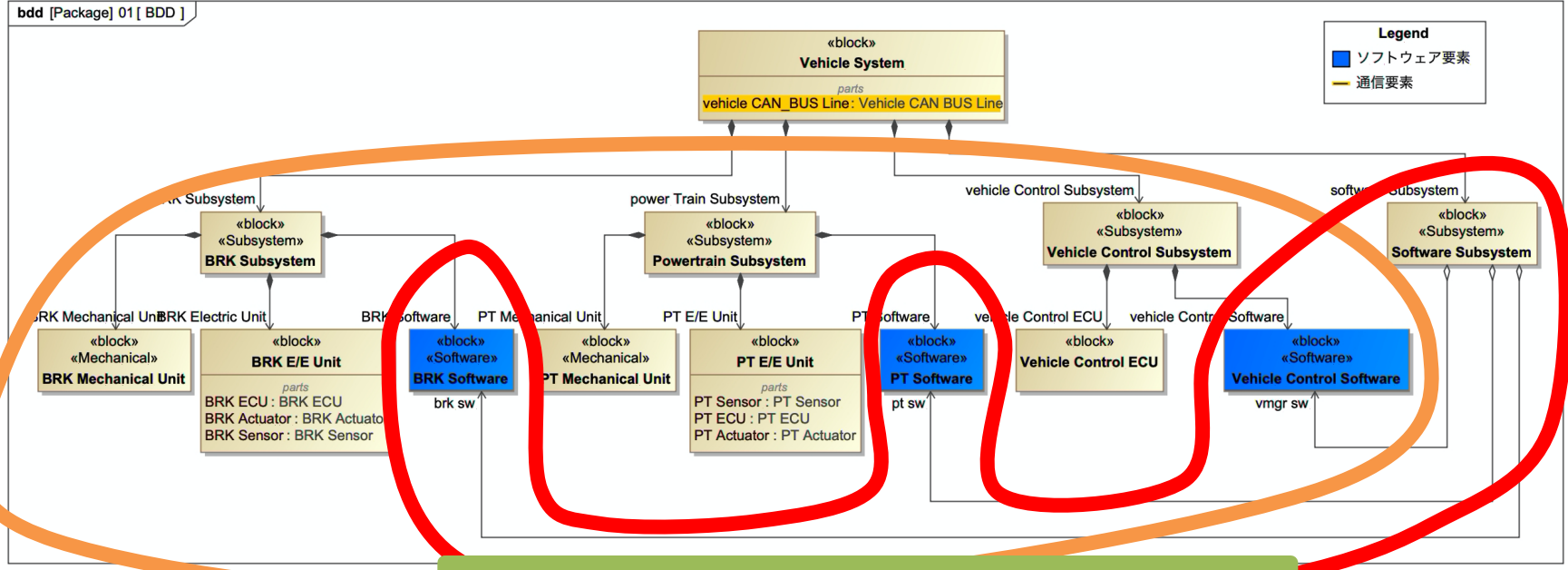
①-2情報の構造化例



①-2情報の構造化例



①-2情報の構造化例



開発に貢献するシステム記述(システムモデル)

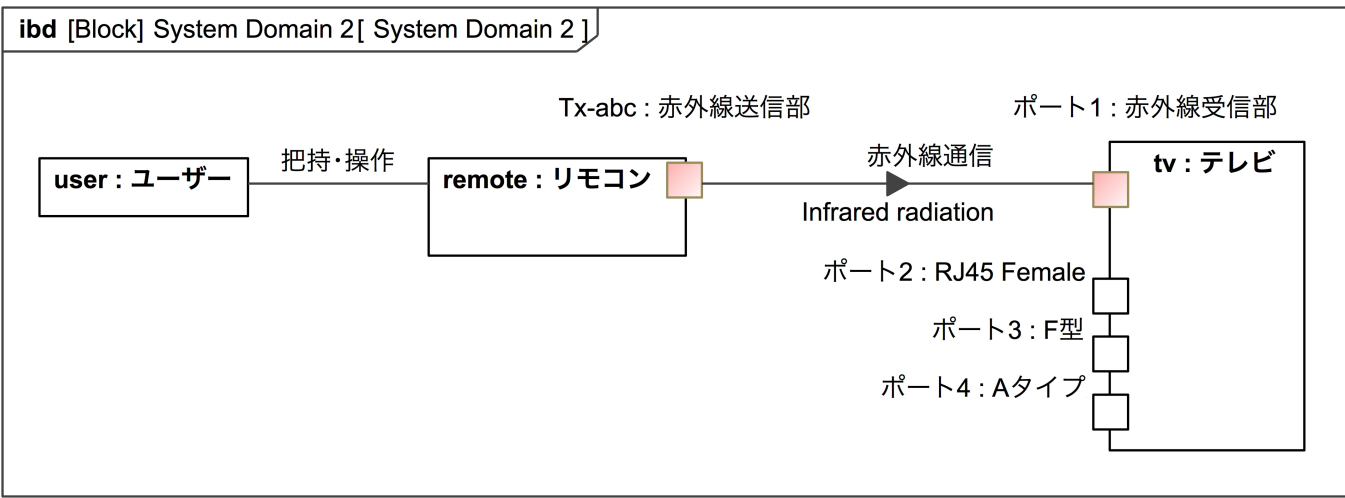
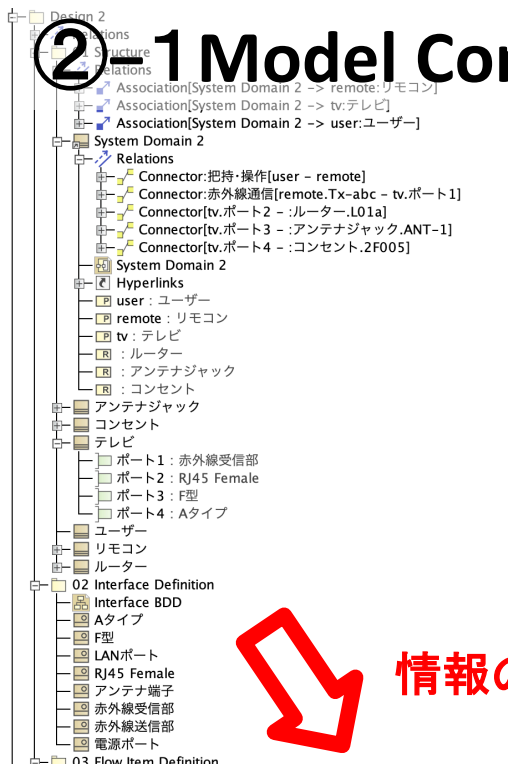
コツ②

**Modeling(モデリング)に注力、ではなく、
Model Consumption(モデル活用)に注力**

コツ②Modeling(モデリング)に注力、ではなく、 Model Consumption(モデル活用)に注力

- システム開発をうまく進めるために、情報構造体から任意の情報を、任意の形態で抽出する。

②-1 Model Consumption(モデル活用)例



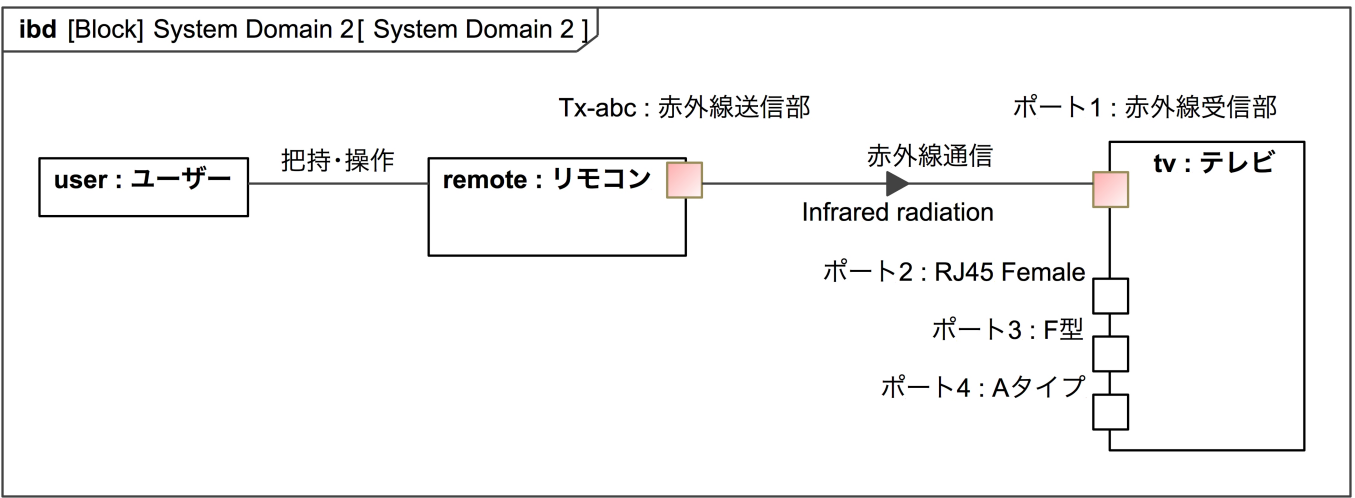
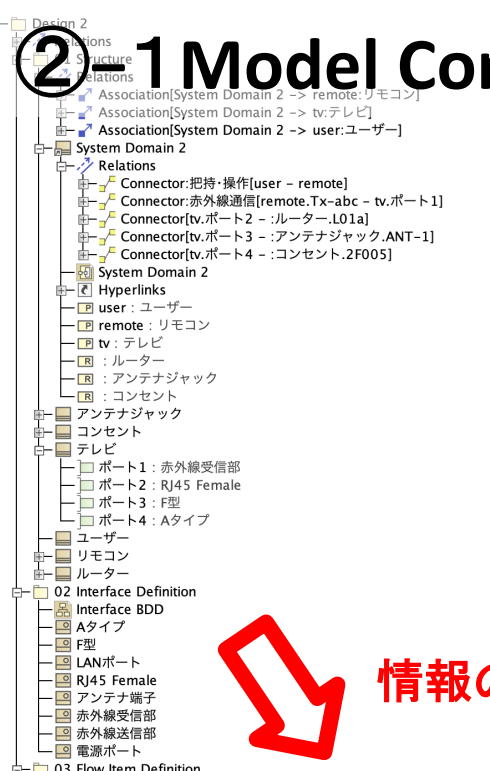
情報の抽出とエンジニアリングでの利用(=活用)

#	△ Name	Type	接続	接続ポート	接続先
1	ポート1	赤外線受信部	Connector:赤外線通信[remote.Tx-abc - tv.ポート1]	Tx-abc : 赤外線送信部	リモコン
2	ポート2	RJ45 Female	Connector[tv.ポート2 - :ルーター.L01a]	L01a : ~LANポート	ルーター
3	ポート3	F型	Connector[tv.ポート3 - :アンテナジャック.ANT-1]	ANT-1 : ~アンテナ端子	アンテナジャック
4	ポート4	Aタイプ	Connector[tv.ポート4 - :コンセント.2F005]	2F005 : ~電源ポート	コンセント

コツ②Modeling(モデリング)に注力、ではなく、 Model Consumption(モデル活用)に注力

- Modeling(モデリング)＝情報の構造化
- Model Consumption(モデル活用)＝情報構造体の活用

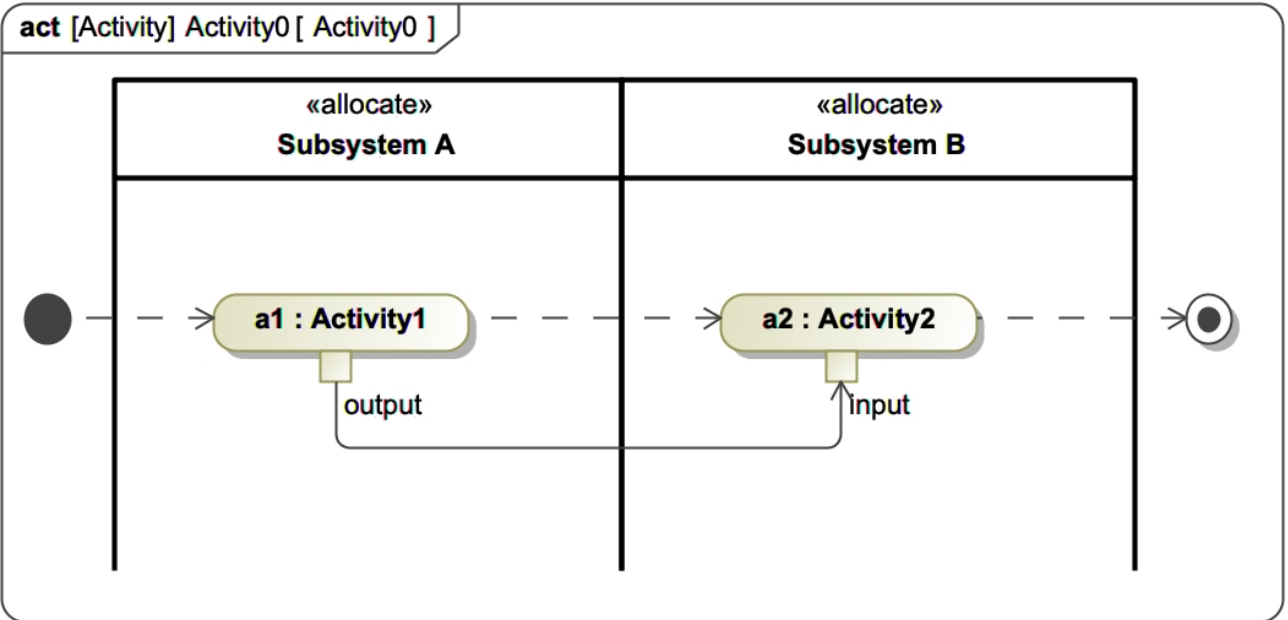
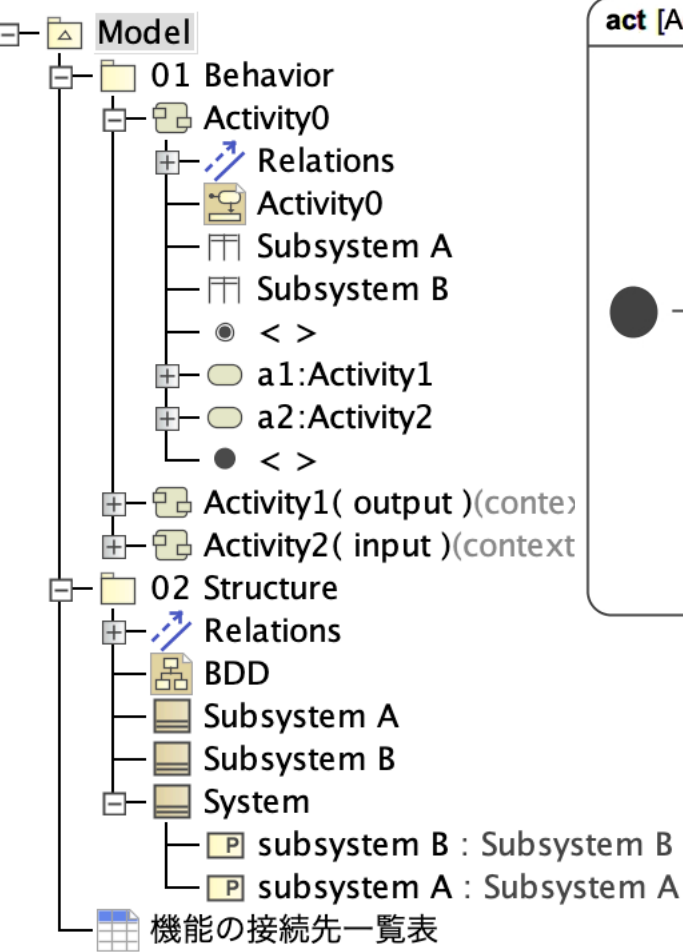
②-1 Model Consumption(モデル活用)例



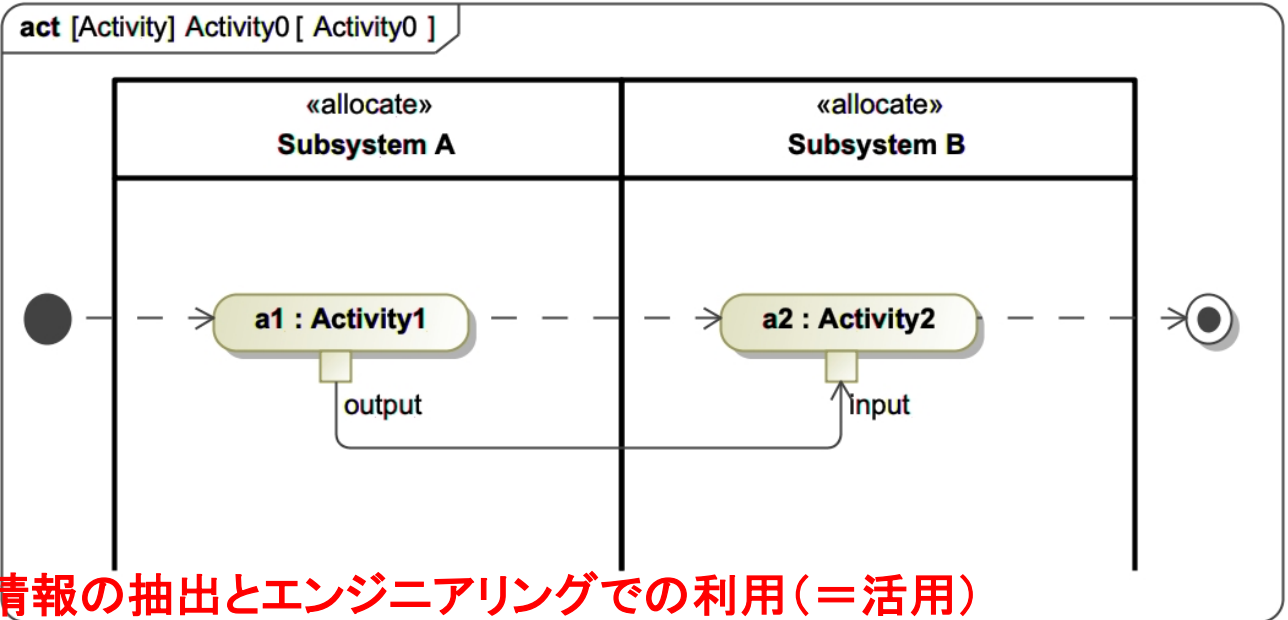
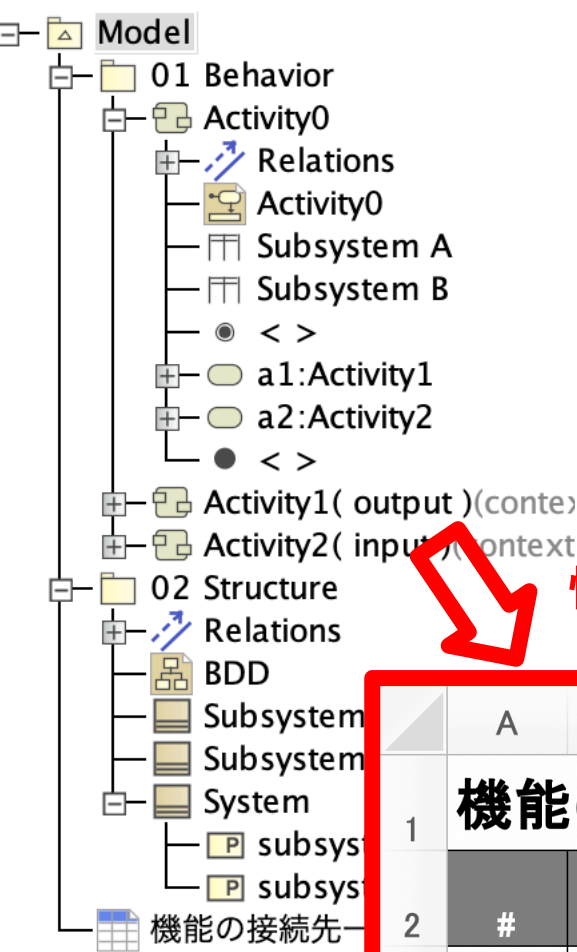
情報の抽出とエンジニアリングでの利用(＝活用)

#	△ Name	Type	接続	接続ポート	接続先
1	ポート1	赤外線受信部	Connector:赤外線通信[remote.Tx-abc - tv.ポート1]	Tx-abc : 赤外線送信部	リモコン
2	ポート2	RJ45 Female	Connector[tv.ポート2 - :ルーター.L01a]	L01a : ~LANポート	ルーター
3	ポート3	F型	Connector[tv.ポート3 - :アンテナジャック.ANT-1]	ANT-1 : ~アンテナ端子	アンテナジャック
4	ポート4	Aタイプ	Connector[tv.ポート4 - :コンセント.2F005]	2F005 : ~電源ポート	コンセント

②-2 Model Consumption (モデル活用) 例



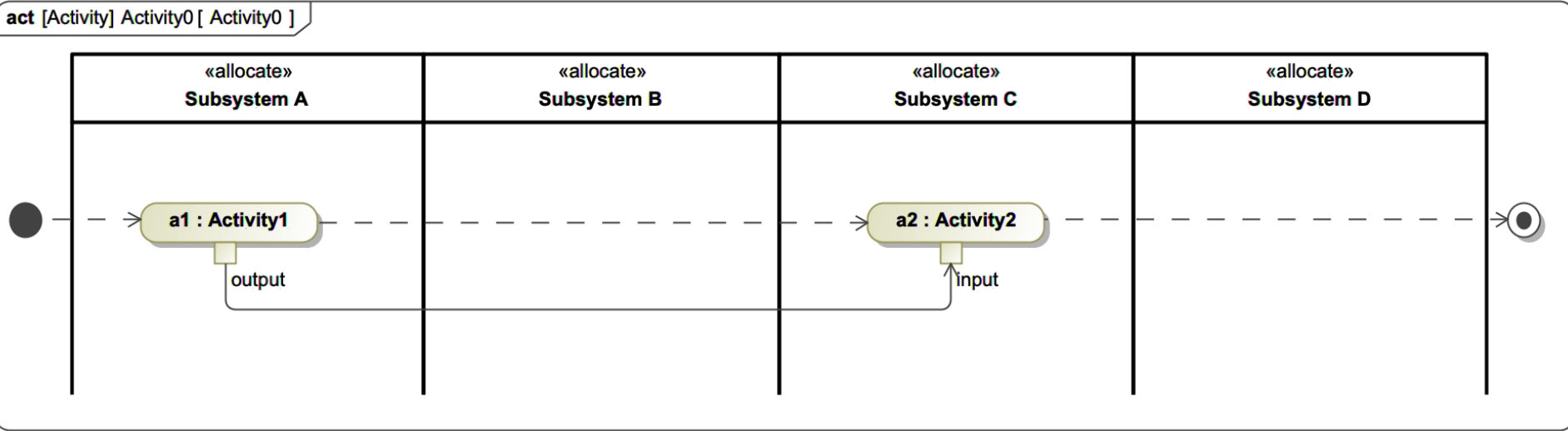
②-2 Model Consumption (モデル活用) 例



情報の抽出とエンジニアリングでの利用(=活用)

	A	B	<<Excel ファイル>>	C
1	機能の接続先一覧表			
2	#	Name	機能の接続先サブシステム	
3	1	Subsystem A	Subsystem B	
4				
5				
6				
7				

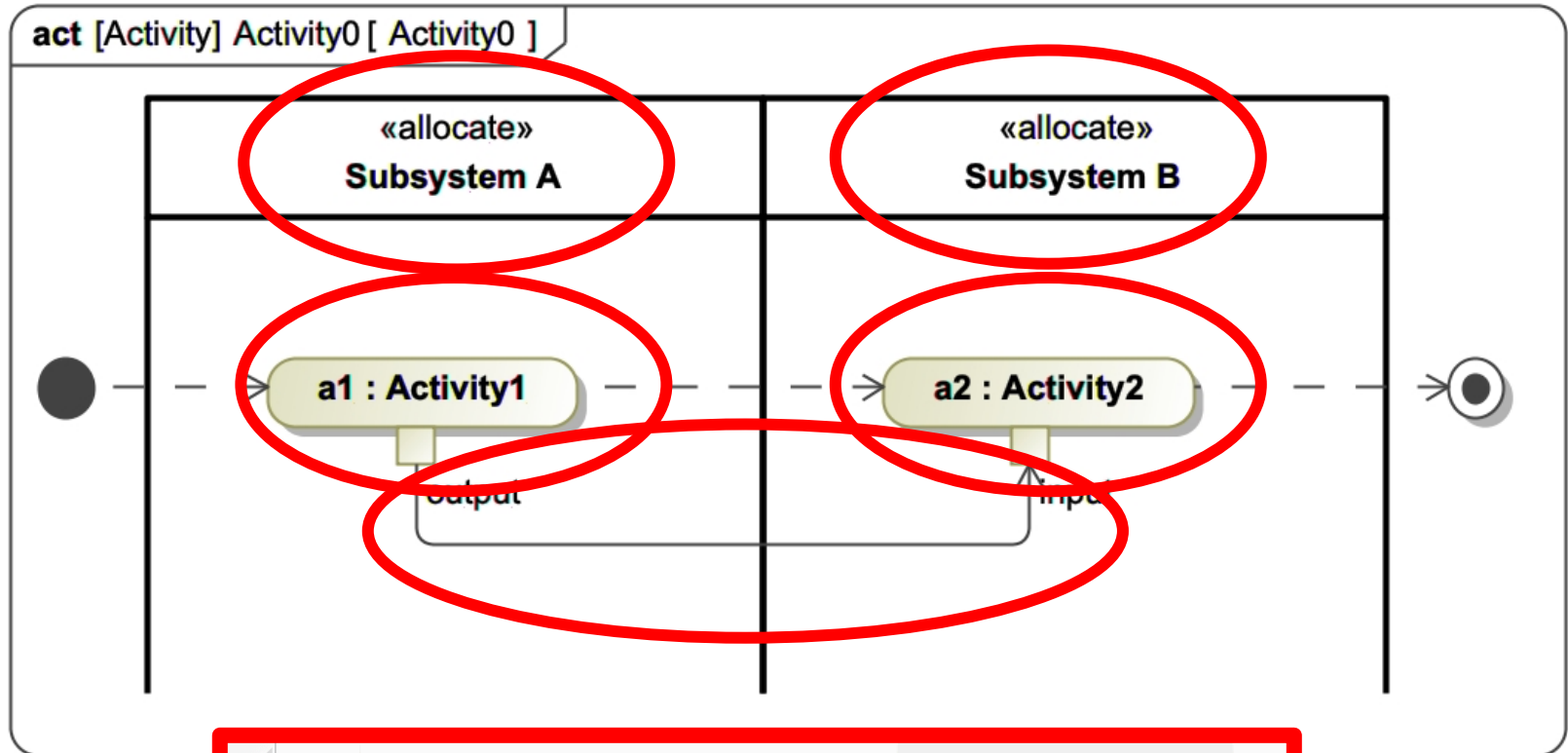
②-2 Model Consumption (モデル活用) 例



	A	B	C	
1	機能の接続先一覧表			
2	#	Name	機能の接続先サブシステム	
3	1	Subsystem A	Subsystem C	
4				
5				
6				
7				

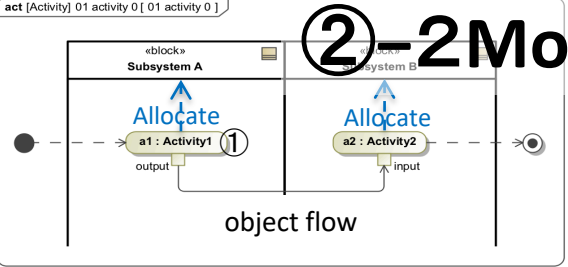
②-2 Model Consumption (モデル活用) 例

“メタ構造”を辿り、情報を“クエリ”する技術

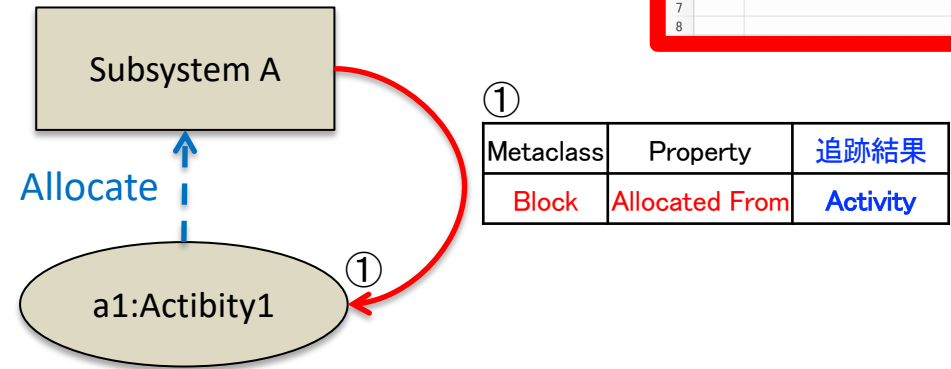


	A	B	C
1	Model		
2	#	Name	機能の接続先サブシステム
3	1	Subsystem A	Subsystem B
4			
5			
6			
7			
8			

②-2 Model Consumption (モデル活用) 例



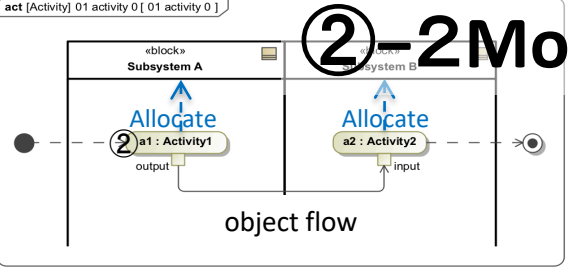
	A	B	C
1	Model		
2	#	Name	機能の接続先サブシステム
3	1	Subsystem A	Subsystem B
4			
5			
6			
7			
8			



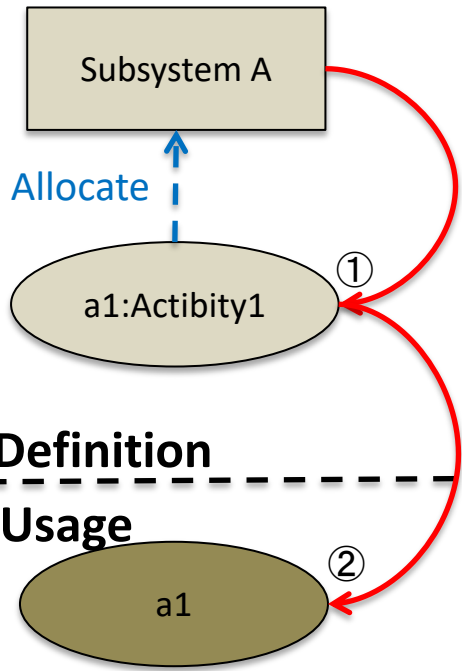
Definition

Usage

②-2 Model Consumption (モデル活用) 例



	A	B	C
1	Model		
2	#	Name	機能の接続先サブシステム
3	1	Subsystem A	Subsystem B
4			
5			
6			
7			
8			



①

Metaclass	Property	追跡結果
Block	Allocated From	Activity

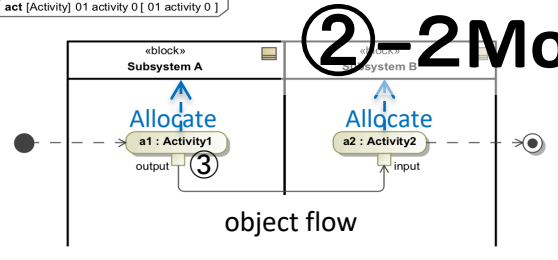
②

Metaclass	Property	追跡結果
Activity	CallBehaviorAction OfBehavior	CallBehavior Action

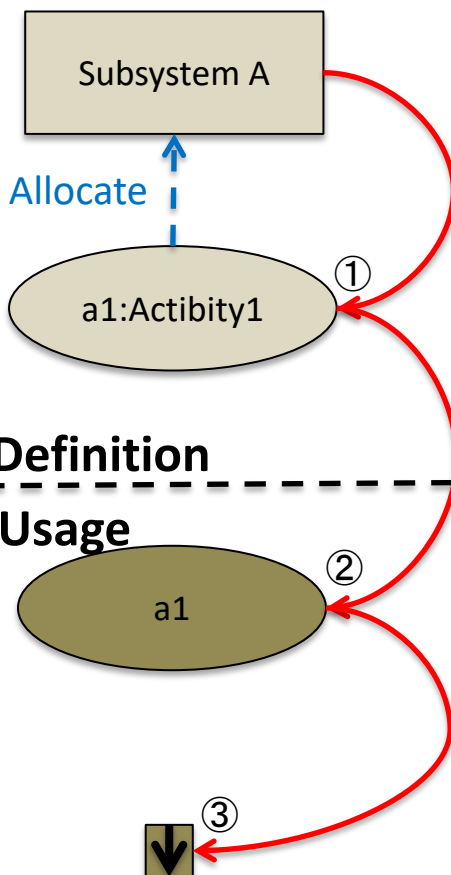
Definition

Usage

②-2 Model Consumption (モデル活用) 例



	A	B	C
1	Model		
2	#	Name	機能の接続先サブシステム
3	1	Subsystem A	Subsystem B
4			
5			
6			
7			
8			



①

Metaclass	Property	追跡結果
Block	Allocated From	Activity

②

Metaclass	Property	追跡結果
Activity	CallBehaviorAction OfBehavior	CallBehavior Action

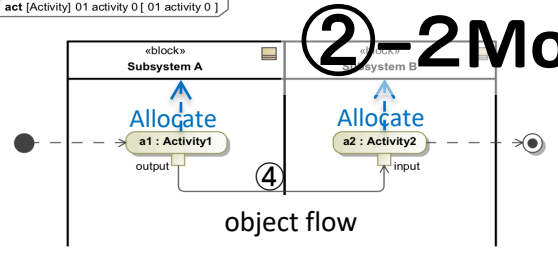
③

Metaclass	Property	追跡結果
Action	Output	OutputPin

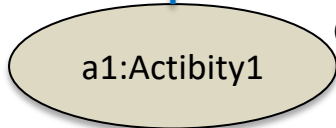
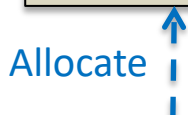
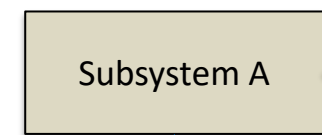
Definition

Usage

②-2 Model Consumption (モデル活用) 例



	A	B	C
1	Model		
2	#	Name	機能の接続先サブシステム
3	1	Subsystem A	Subsystem B
4			
5			
6			
7			
8			



①

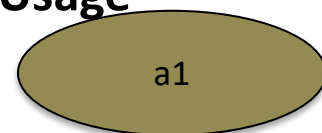
Metaclass	Property	追跡結果
Block	Allocated From	Activity

②

Metaclass	Property	追跡結果
Activity	CallBehaviorAction OfBehavior	CallBehavior Action

Definition

Usage



③

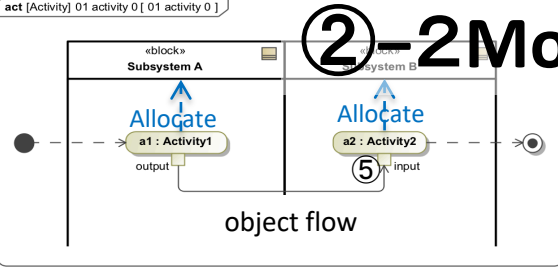
Metaclass	Property	追跡結果
Action	Output	OutputPin

④

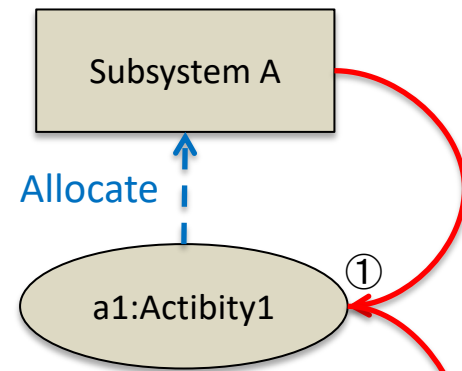
Metaclass	Property	追跡結果
OutputPin	Outgoing	ObjectFlow



②-2 Model Consumption (モデル活用) 例



A	B	C
Model		
1		
2	#	Name
3	1	Subsystem A
4		機能の接続先サブシステム
5		Subsystem B
6		
7		
8		



①

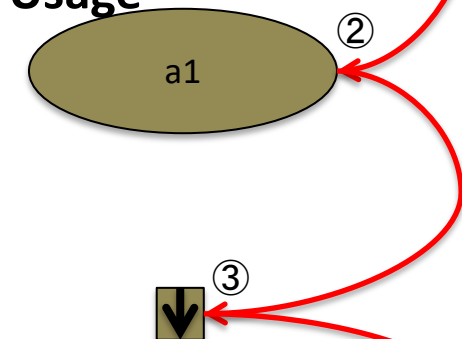
Metaclass	Property	追跡結果
Block	Allocated From	Activity

②

Metaclass	Property	追跡結果
Activity	CallBehaviorAction OfBehavior	CallBehavior Action

Definition

Usage



③

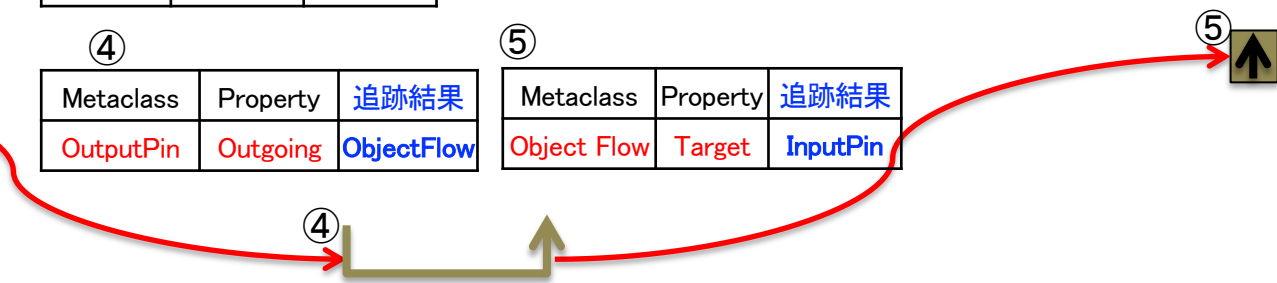
Metaclass	Property	追跡結果
Action	Output	OutputPin

④

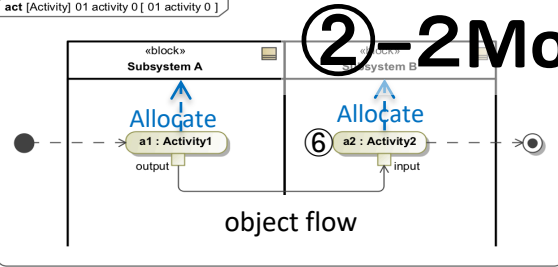
Metaclass	Property	追跡結果
OutputPin	Outgoing	ObjectFlow

⑤

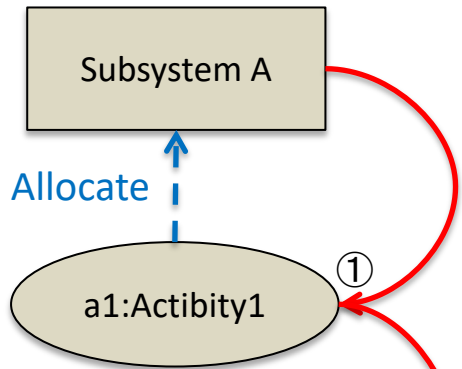
Metaclass	Property	追跡結果
Object Flow	Target	InputPin



②-2 Model Consumption (モデル活用) 例



A	B	C
Model		
1		
2	#	Name
3	1	Subsystem A
4		
5		
6		
7		
8		



①

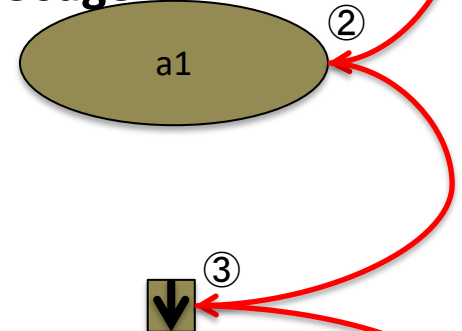
Metaclass	Property	追跡結果
Block	Allocated From	Activity

②

Metaclass	Property	追跡結果
Activity	CallBehaviorAction OfBehavior	CallBehavior Action

Definition

Usage

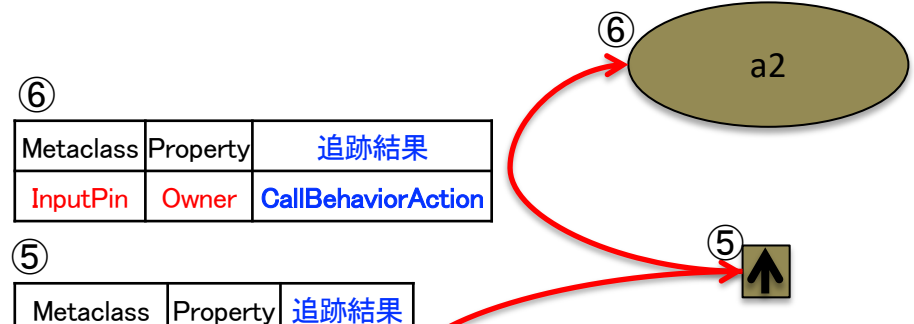


③

Metaclass	Property	追跡結果
Action	Output	OutputPin

④

Metaclass	Property	追跡結果
OutputPin	Outgoing	ObjectFlow



⑥

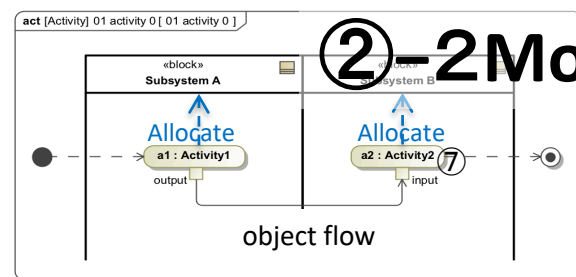
Metaclass	Property	追跡結果
InputPin	Owner	CallBehaviorAction

⑤

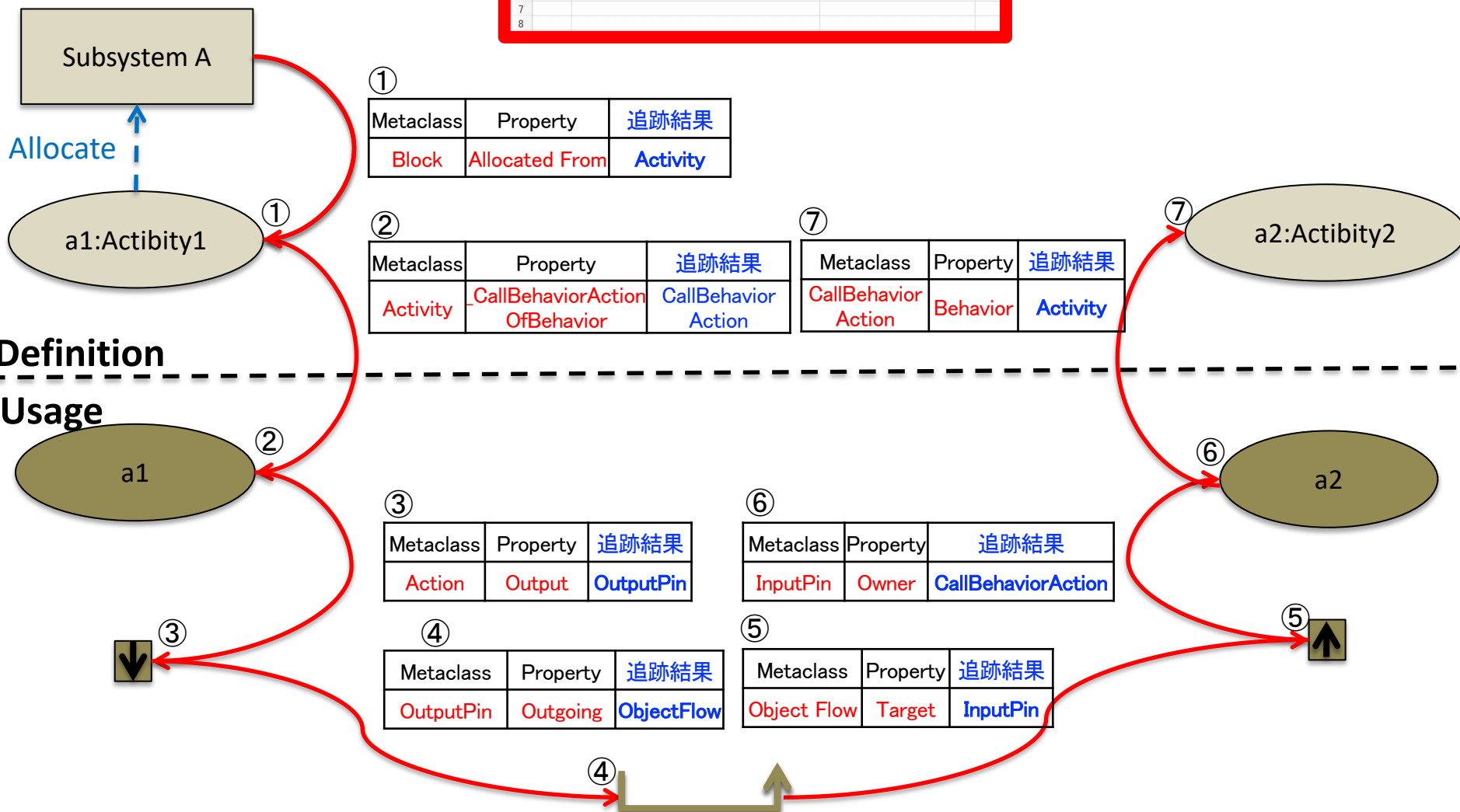
Metaclass	Property	追跡結果
Object Flow	Target	InputPin



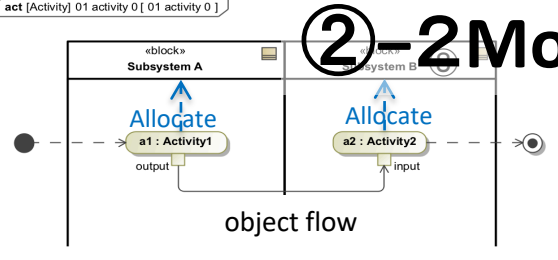
②-2 Model Consumption (モデル活用) 例



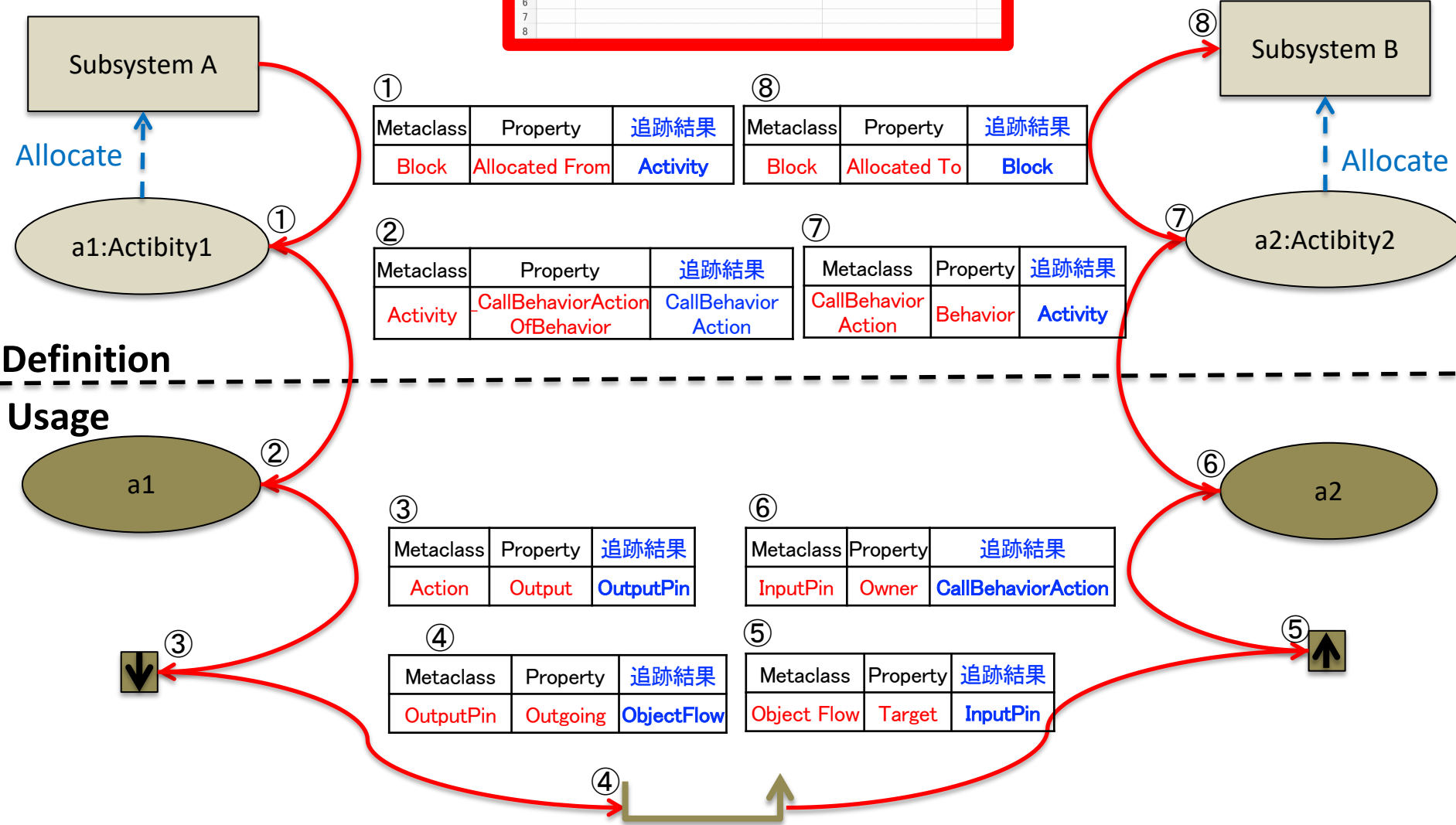
A	B	C
Model		
1		
2	#	Name
3	1	Subsystem A
4		機能の接続先サブシステム
5		Subsystem B
6		
7		
8		



②-2 Model Consumption (モデル活用) 例



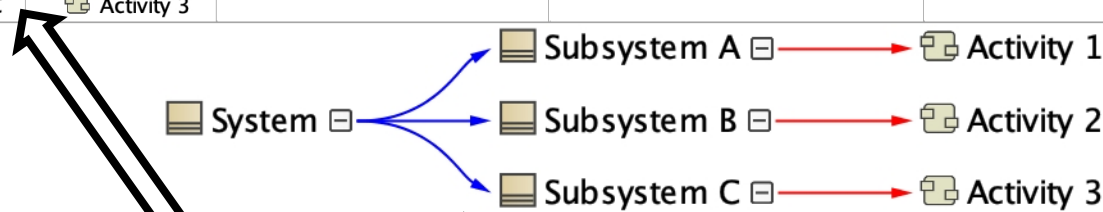
A	B	C
Model		
1		
2	#	Name
3	1	Subsystem A
4		機能の接続先サブシステム
5		Subsystem B
6		
7		
8		



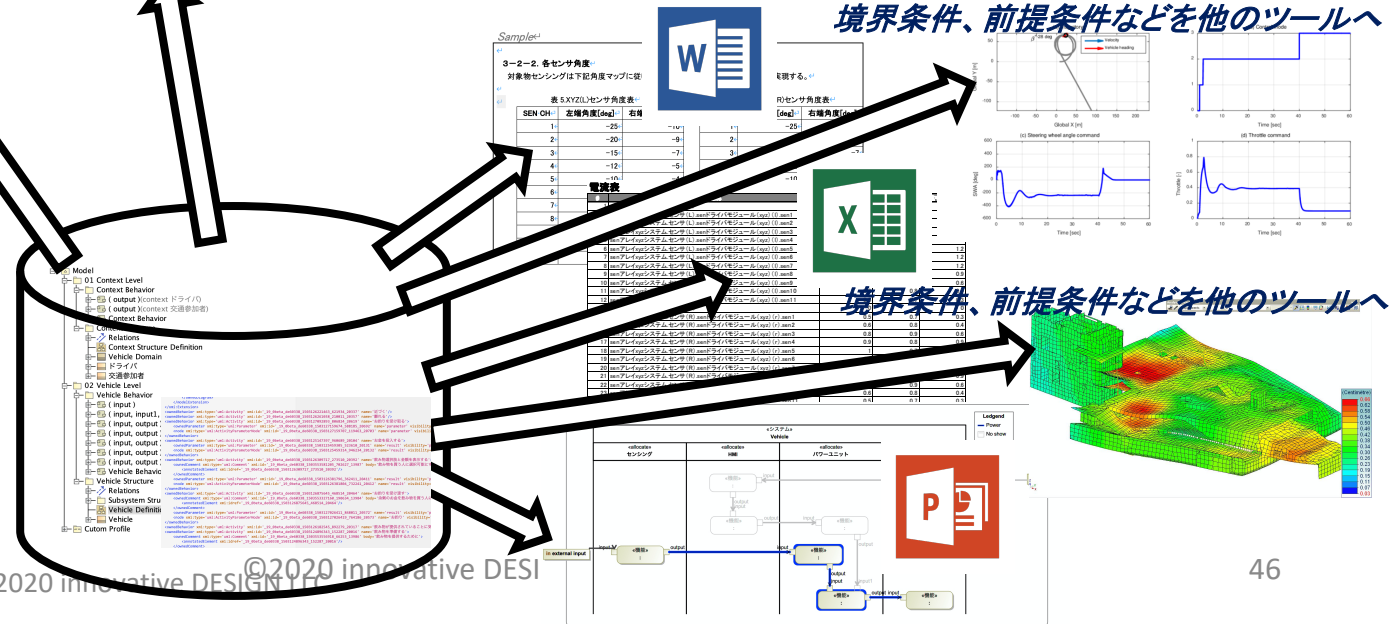
コツ②Modeling(モデリング)に注力、ではなく、 Model Consumption(モデル活用)に注力

- システム開発をうまく進めるために、情報構造体から任意の情報を、任意の形態で抽出する。

#	出力サブシステム	△ Name	出力元	Object Flow	入力先	機能	入力先サブシステム
1	Subsystem A	Activity 1	output : Air output1 : High Pressure Air output2 : Low Presssure Air	Object Flow[output -> input] Object Flow[output1 -> input1] Object Flow[output2 -> input2]	input : Air input1 : Air input2 : Low Presssure Air	Activity 2(context Sub	Subsystem B
2	Subsystem B	Activity 2	output	Object Flow[output -> input]	input	Activity 3(context Sub	Subsystem C
3	Subsystem C	Activity 3					



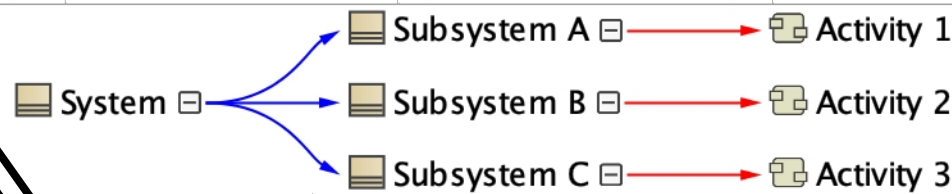
情報構造体



コツ②Modeling(モデリング)に注力、ではなく、 Model Consumption(モデル活用)に注力

- システム開発をうまく進めるために、情報構造体から任意の情報を、任意の形態で抽出する。

#	出力サブシステム	△ Name	出力元	Object Flow	入力先	機能	入力先サブシステム
1	Subsystem A	Activity 1	output : Air output1 : High Pressure Air output2 : Low Presssure Air	Object Flow[output -> input] Object Flow[output1 -> input1] Object Flow[output2 -> input2]	input : Air input1 : Air input2 : Low Presssure Air	Activity 2(context Sub	Subsystem B
2	Subsystem B	Activity 2	output	Object Flow[output -> input]	input	Activity 3(context Sub	Subsystem C
3	Subsystem C	Activity 3					



情報構造体

境界条件、前提条件などを他のツールへ

境界条件、前提条件などを他のツールへ

システム記述言語の構造を
理解したクエリ技術が重要

開発に貢献するシステム記述(システムモデル)

コツ③

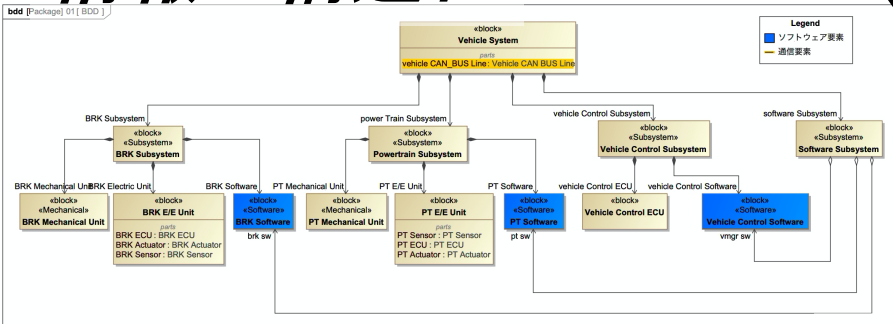
可視化、ではなく、問題・課題を解決する

コツ③可視化、ではなく、問題・課題を解決する

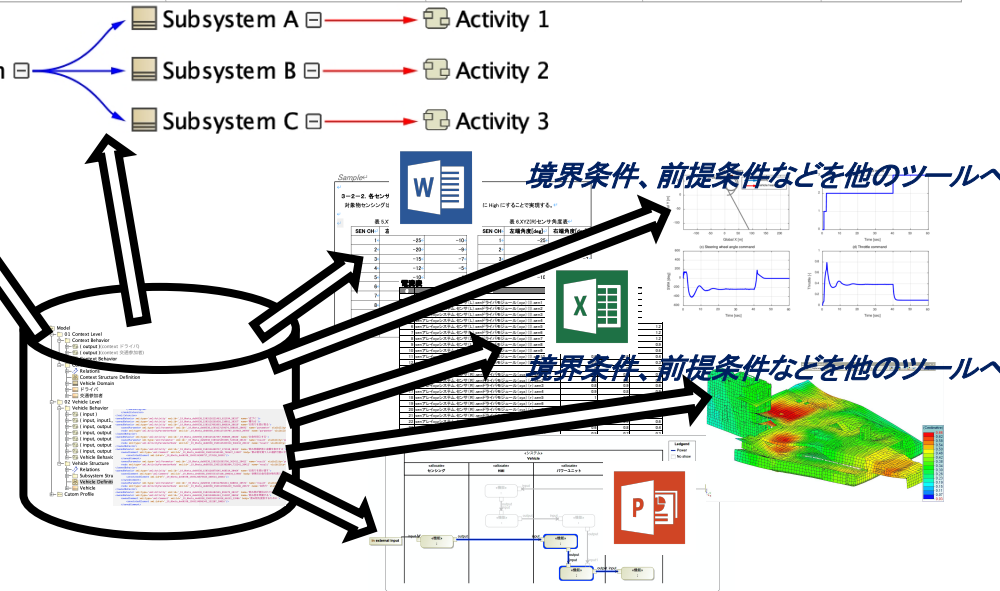
- システム開発を進める中で、システムズエンジニアが直面する問題や課題の解決のために、「情報の構造化」と「Model Consumption(モデル活用)」も駆使する。

#	出力サブシステム	△ Name	出力元	Object Flow	入力先	機能	入力先サブシステム
1	Subsystem A	Activity 1	output : Air output1 : High Pressure Air output2 : Low Presssure Air	Object Flow[output -> input] Object Flow[output1 -> input1] Object Flow[output2 -> input2]	input : Air input1 : Air input2 : Low Presssure Air	Activity 2(context Sub	Subsystem B
2	Subsystem B	Activity 2	output	Object Flow[output -> input]	input	Activity 3(context Sub	Subsystem C
3	Subsystem C	Activity 3					

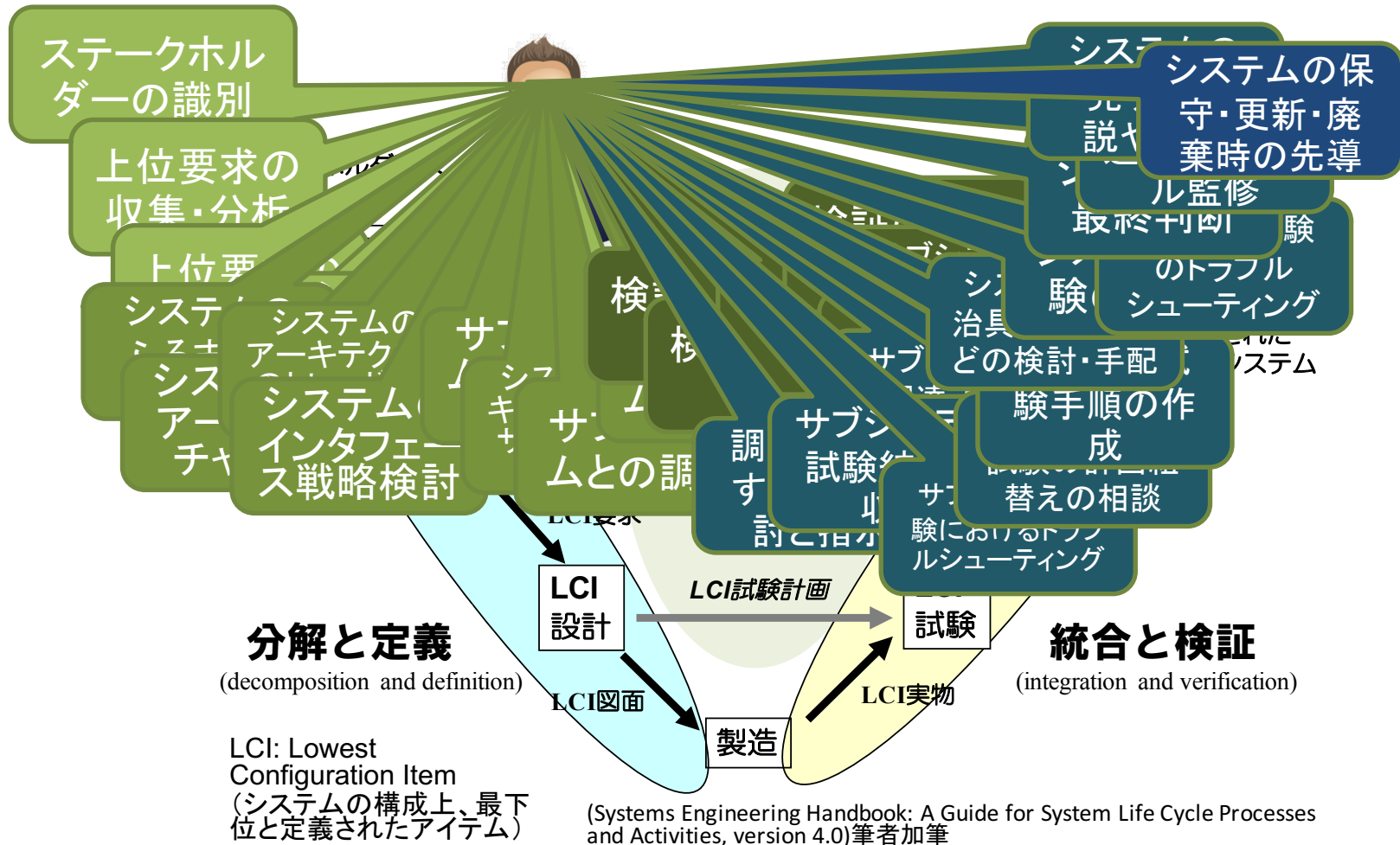
情報の構造化



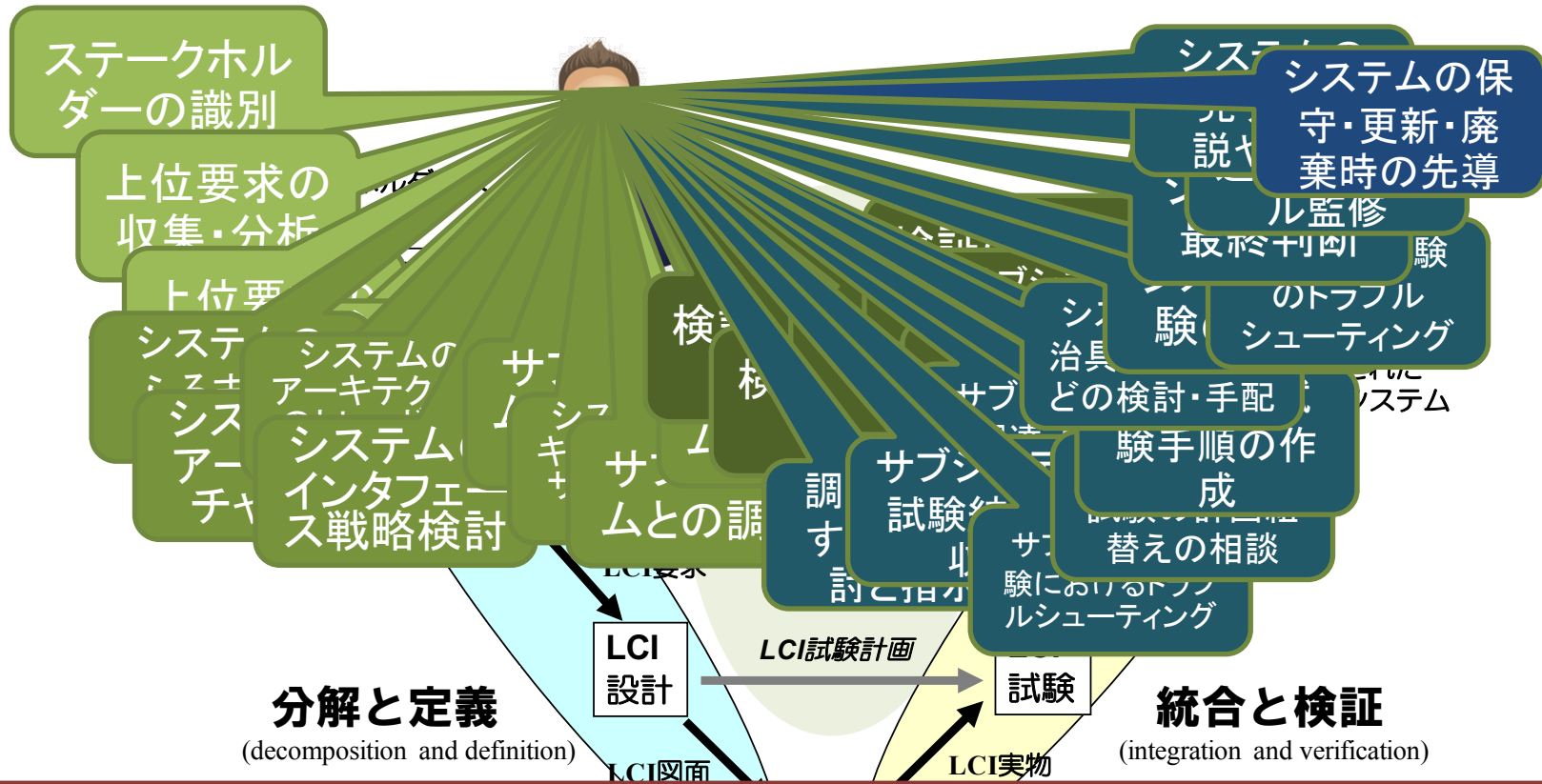
Model Consumption (モデル活用)



システムズエンジニアのイメージ

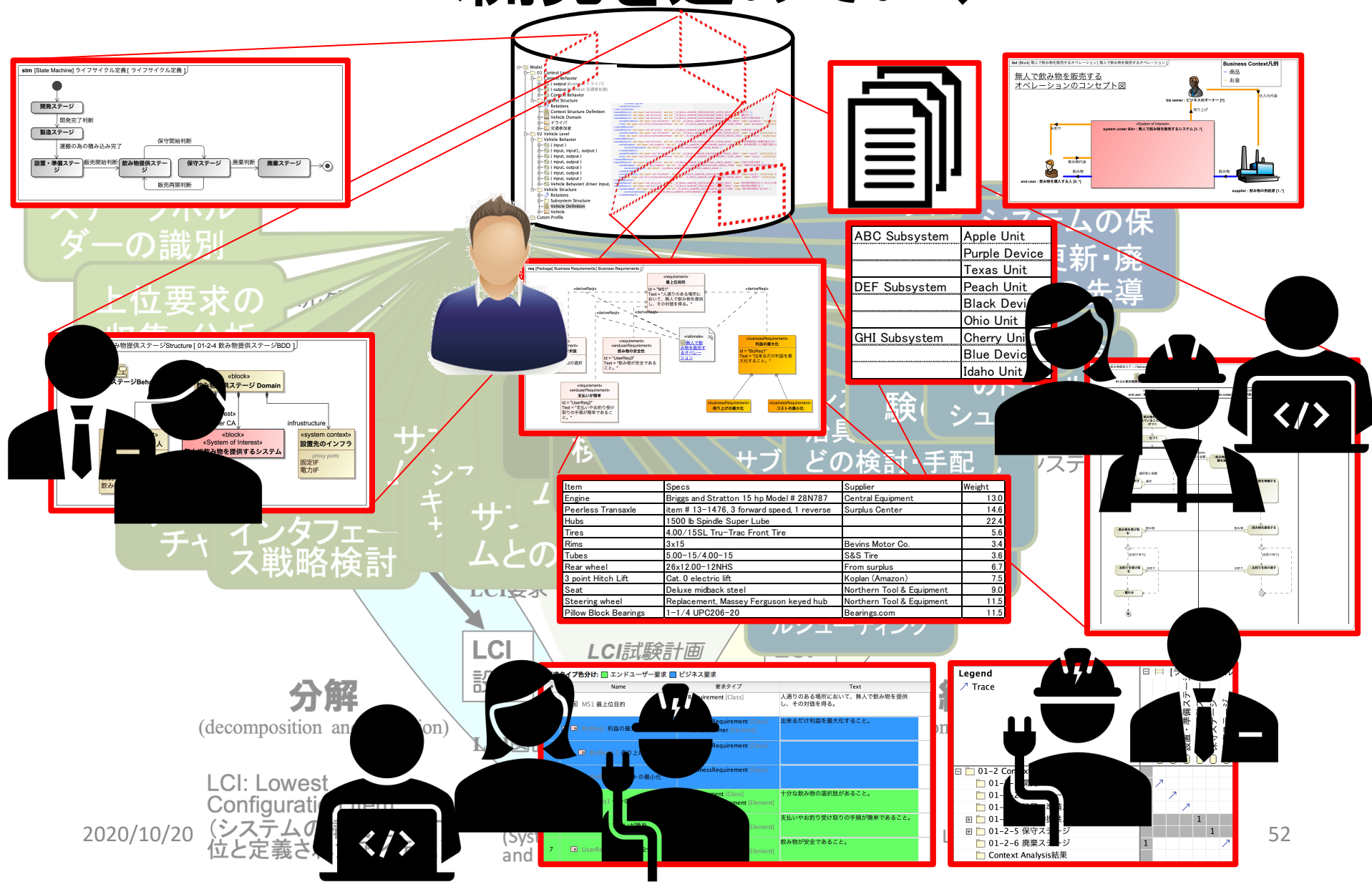


システムズエンジニアのイメージ

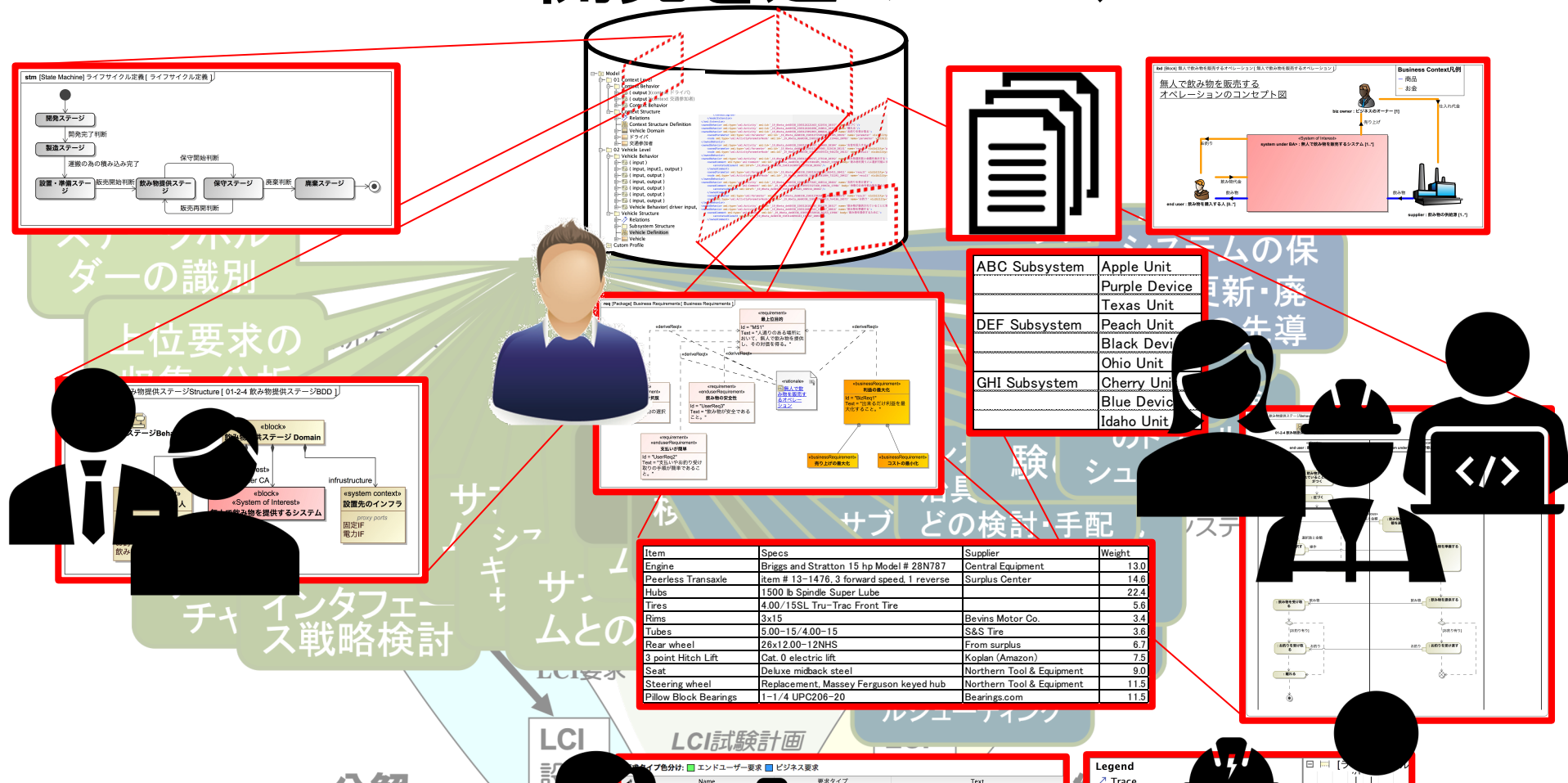


システムズエンジニアは、システムが何の目的で何を果たすのかを明確にし、それがどのような環境や状況下であるのかを丁寧に洗い出し、そのためにシステムに求められる特性・特徴・機能・性能などを明らかにし、その実現手段を広く探索して解を導き、その実現解が確実に実現・実装されるべく考え方とプロセスで手を尽くす、開発のキーパーソンである。

システムズエンジニアがModel Basedにシステム開発を進めていく

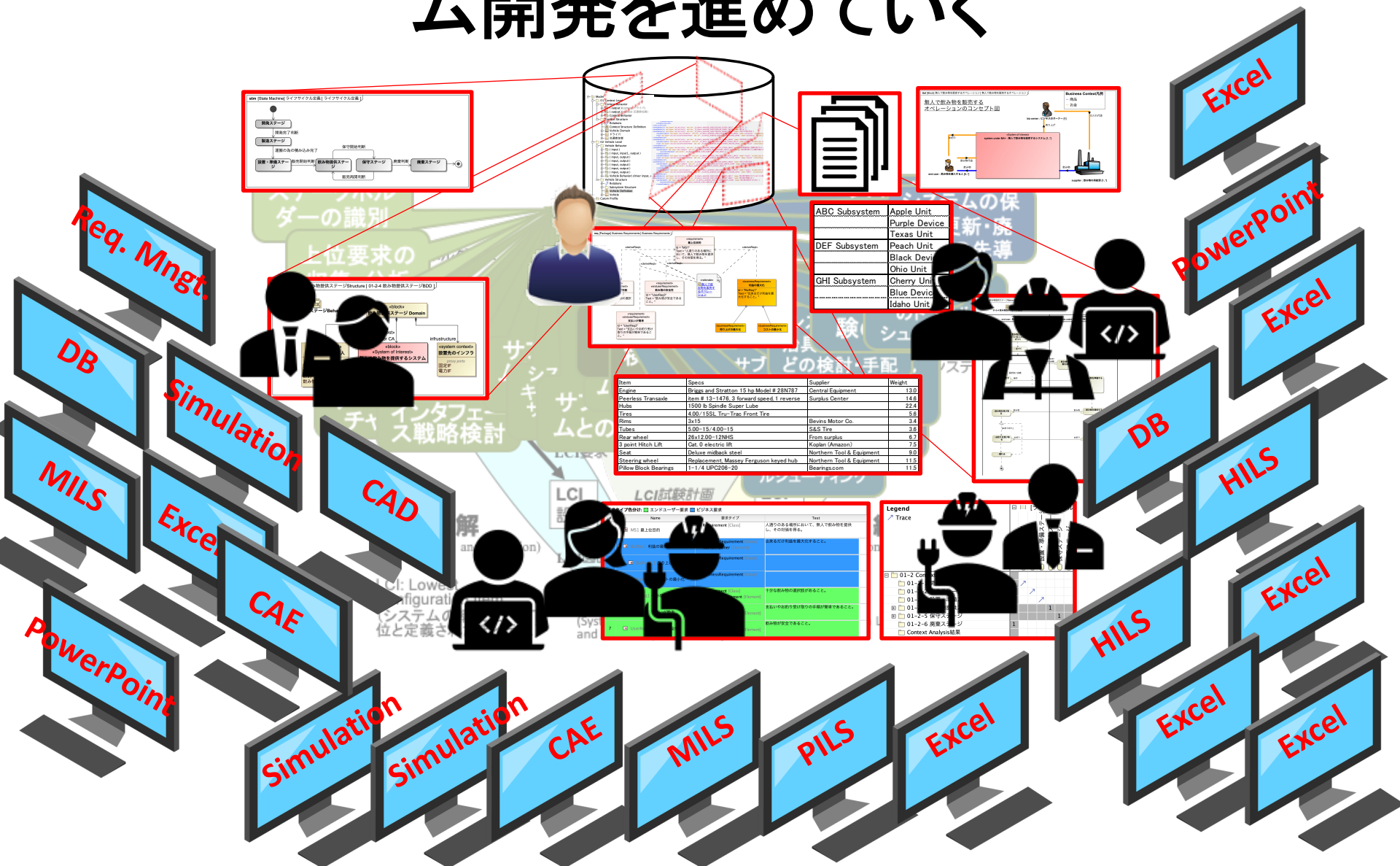


システムズエンジニアがModel Basedにシステム開発を進めていく

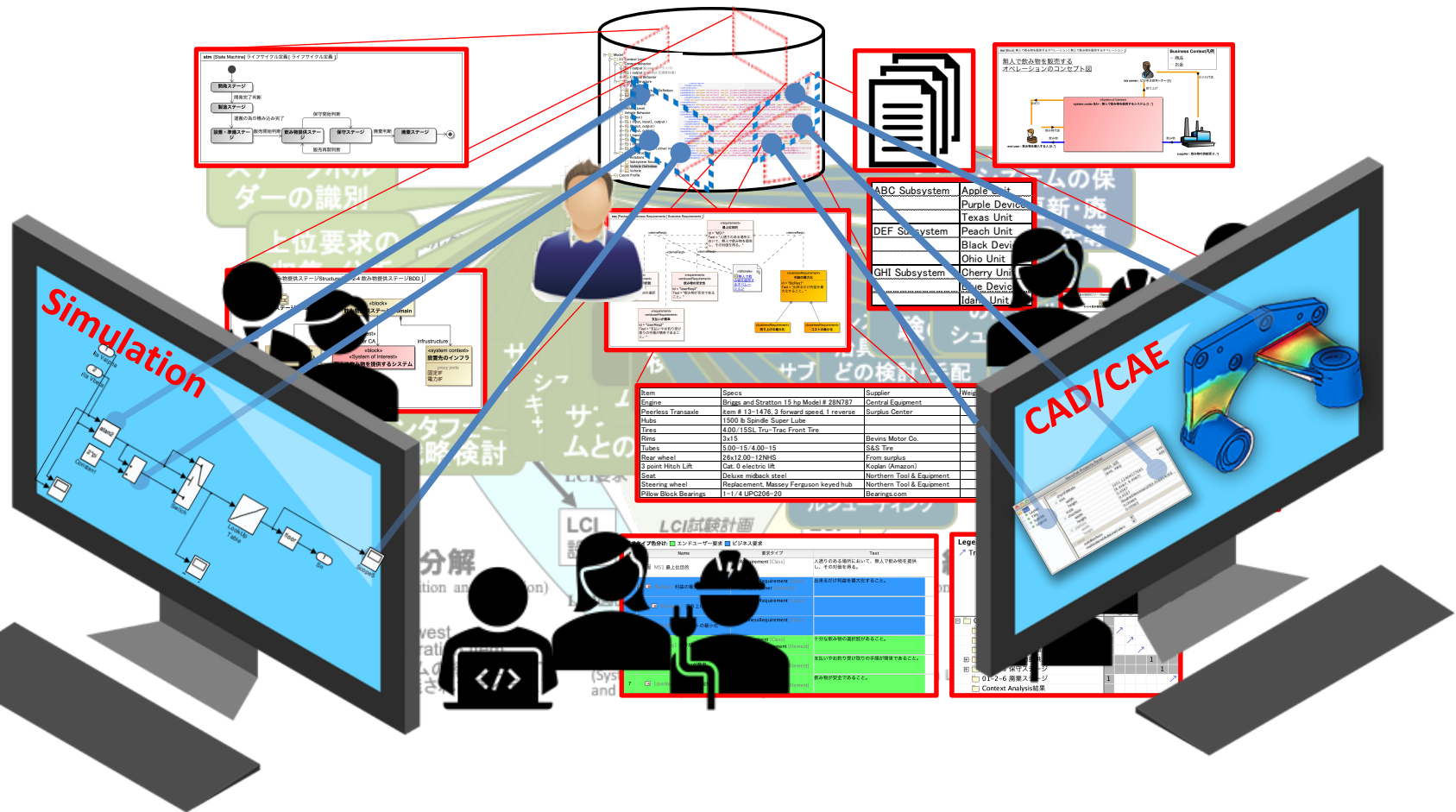


システム開発を進める中で、システムズエンジニアが直面する問題や課題の解決のために、「情報の構造化」と「Model Consumption（モデル活用）」も駆使する。

システムズエンジニアがModel Basedにシステム開発を進めていく



システムズエンジニアがModel Basedにシステム開発を進めていく



システム開発を進める中で、システムズエンジニアが直面する問題や課題の解決のために、「情報の構造化」と「Model Consumption (モデル活用)」も駆使する。

まとめ

- システム記述言語を用いた「システムモデリング」は可視化の手法であると勘違いされることが少なくない。
- システムモデリングとは、システム開発を進めるために必要となる情報を構造化することである。
- システムモデルのModeling: モデリング(=情報の構造化)に注力するのではなく、Model Consumption: モデル活用(=情報構造体の活用)に注力することが重要である。ここでの活用とは、情報の抽出とエンジニアリングでの利用を指す。
- **可視化は決して目的ではない。システム開発における問題・課題を、情報構造体を作成しそれを活用することで解決することが目的である。**

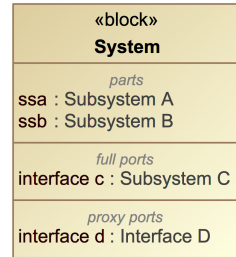
おまけ: システム記述の工夫

SysML (v1.5)を用いた表現力の例示

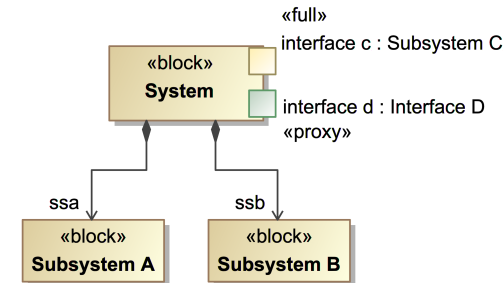
- “開発に役立つ「システム記述」(システムモデリング)の考え方”に基づいた、SysML v1.5を用いた具体的なシステム記述方法を例示する。
 - 本例示においては、SysMLのオーサリングツールとして、CATIA/No MagicのCameo Systems Modeler 19.0LTR SP3を用いている。
-
- ① 【インタフェース記述方法】BlockとPortの構造をより明示的に表した記述方法
 - ② 【インタフェース記述方法】Full PortとProxy Portの記述の使い分け方法
 - ③ 【インタフェース記述方法】Full Portの内部構造を明示的に表した記述方法
 - ④ 【インタフェース記述方法】Full Portのvalue propertyを示し他のpartのvalue propertyと共にParametric Diagramを作成する記述方法
 - ⑤ 【インタフェース記述方法】Proxy Portのflow property定義にGeneralizationによる継承を応用した記述方法
 - ⑥ 【インタフェース記述方法】Item Flowのitem propertyをpart propertyを用いて定義した記述方法
 - ⑦ 【インタフェース記述方法】論理インタフェースと物理インタフェースを表現し分け、割り当て関係を明示するためのitem propertyのallocationの記述方法

①【インタフェース記述方法】BlockとPortの構造をより明示的に表した記述方法

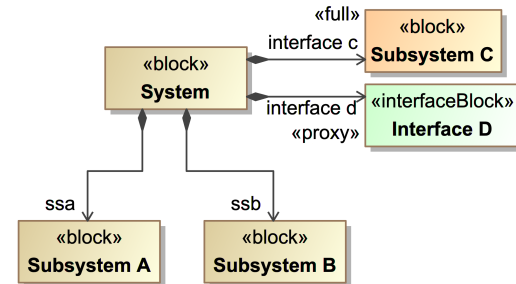
bdd [Package] System Definition 1[System Definition 11]



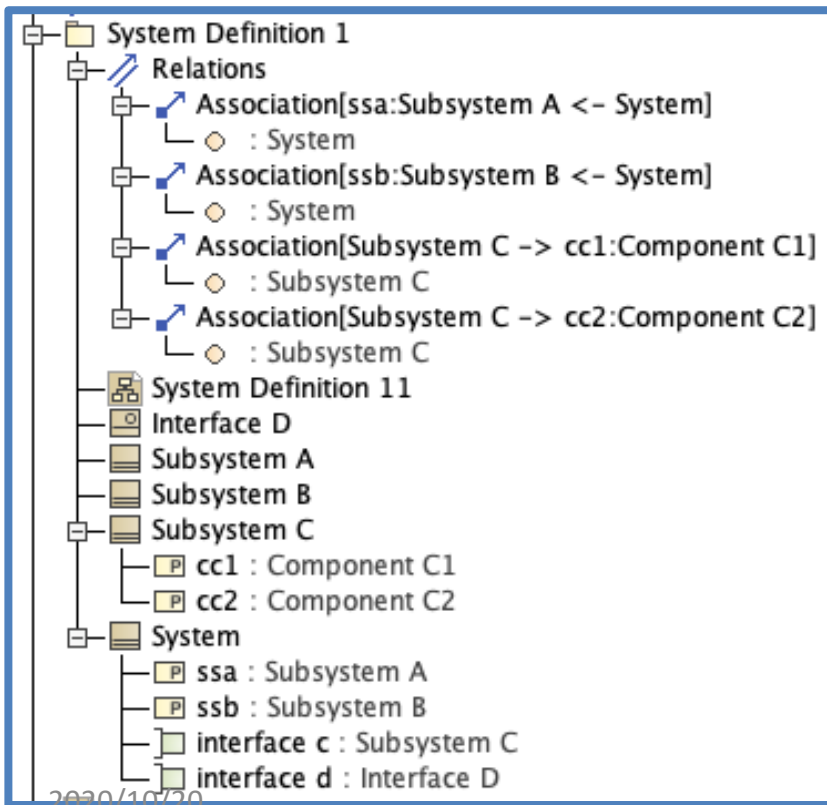
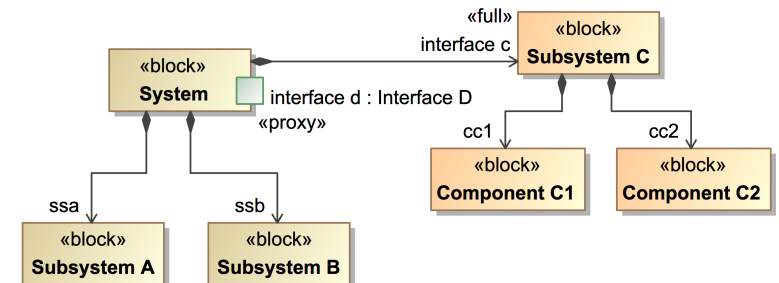
bdd [Package] 01 Structure [System Definition]



bdd [Package] 01 Structure [System Definition 2]



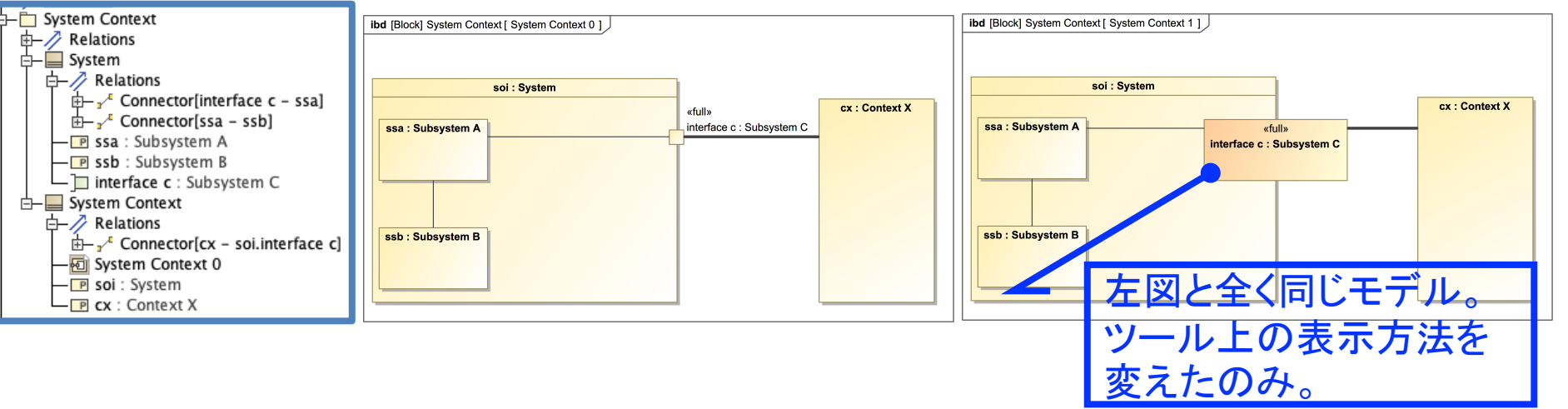
bdd [Package] 01 Structure [System Definition 3]



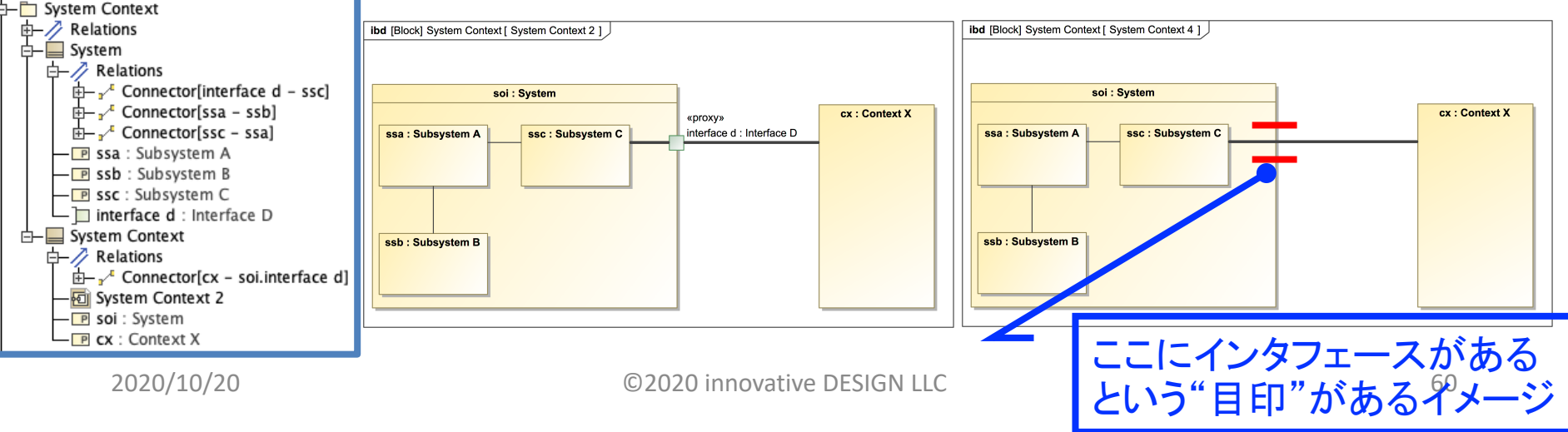
2020/10/20

②【インタフェース記述方法】Full PortとProxy Portの記述の使い分け方法

➤ Full Portとは、Blockの境界に位置するpart propertyであるという考え方。つまり、システムに対するサブシステムの関係性である。

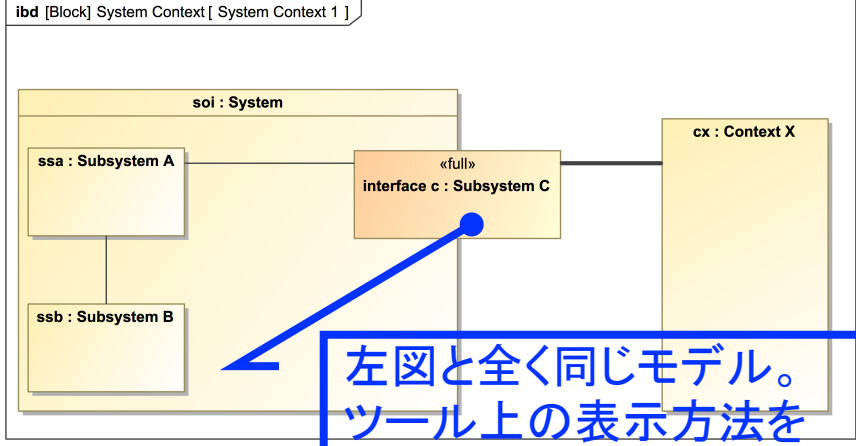
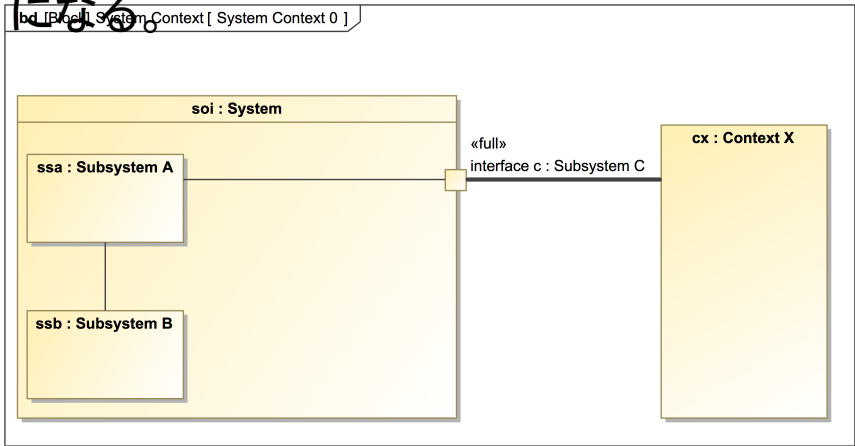


➤ Proxy Portとは、Blockの境界に内部へ接続するインタフェースが存在するという目印を与えているという考え方。つまり、システムに対して実態は無い。

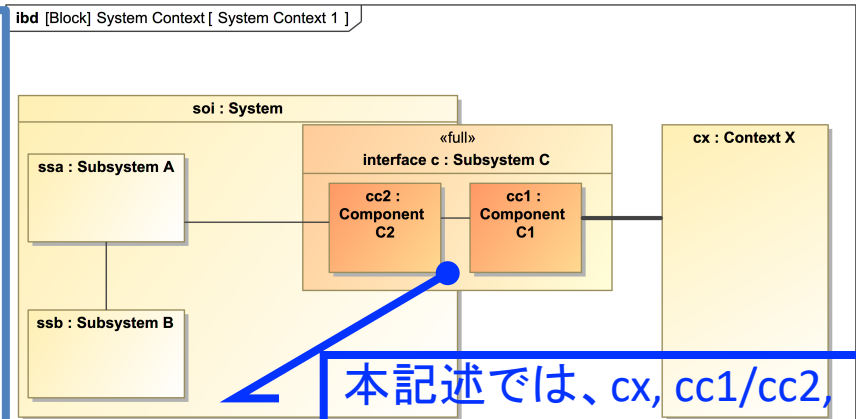
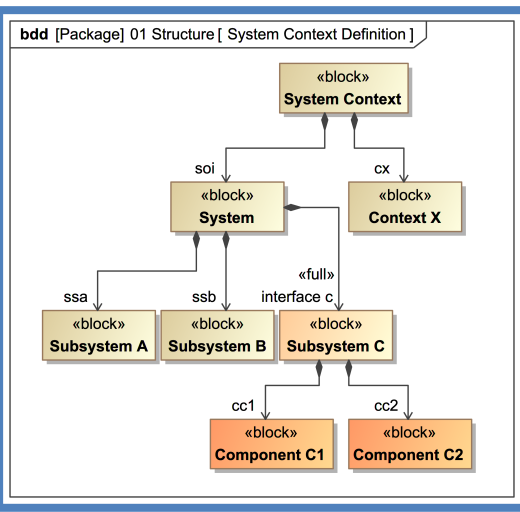
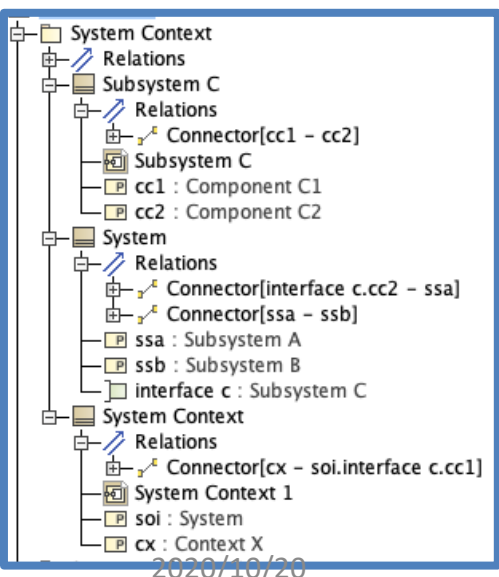


③【インタフェース記述方法】Full Portの内部構造を明示的に表した記述方法

- Full PortのTypeを定義するBlock (本例の場合、Subsystem C)にibdを作成し、「Display Internal Structure」機能で表示することで、意図的に階層を跨いだconnectorの設定も可能になる。

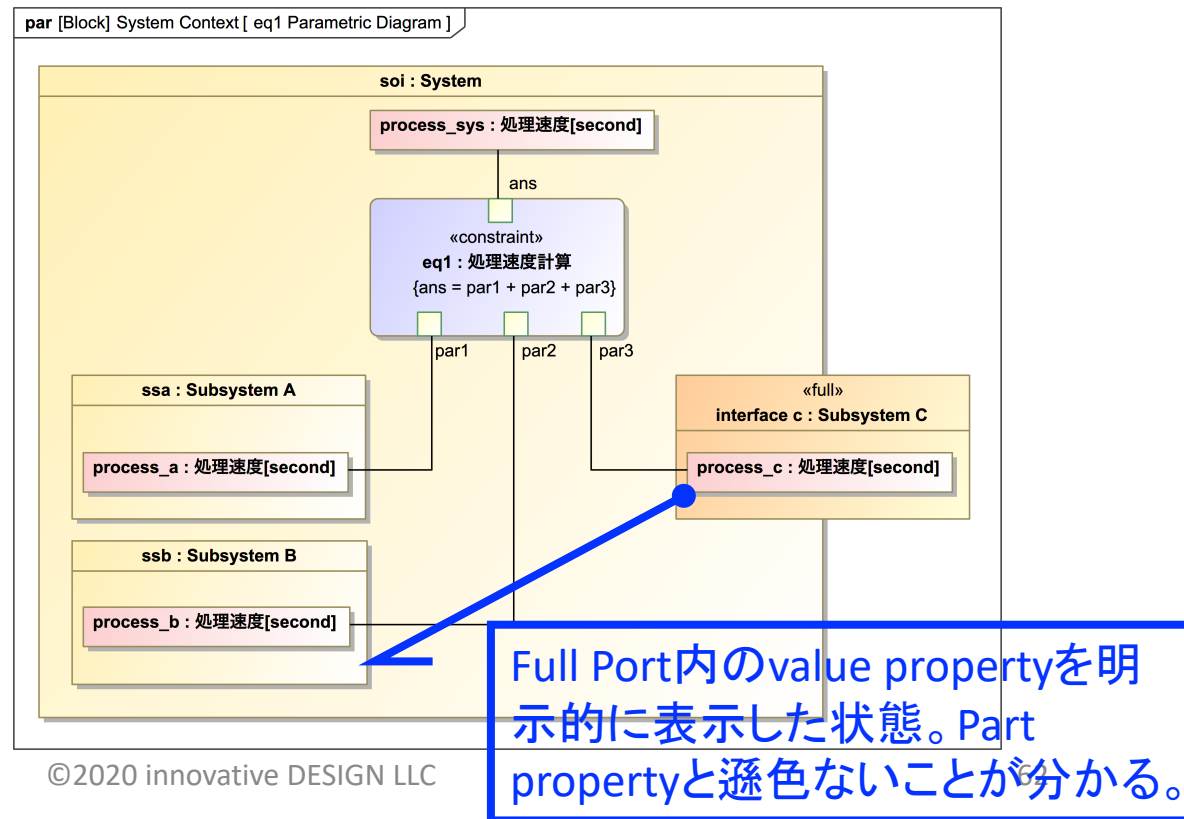
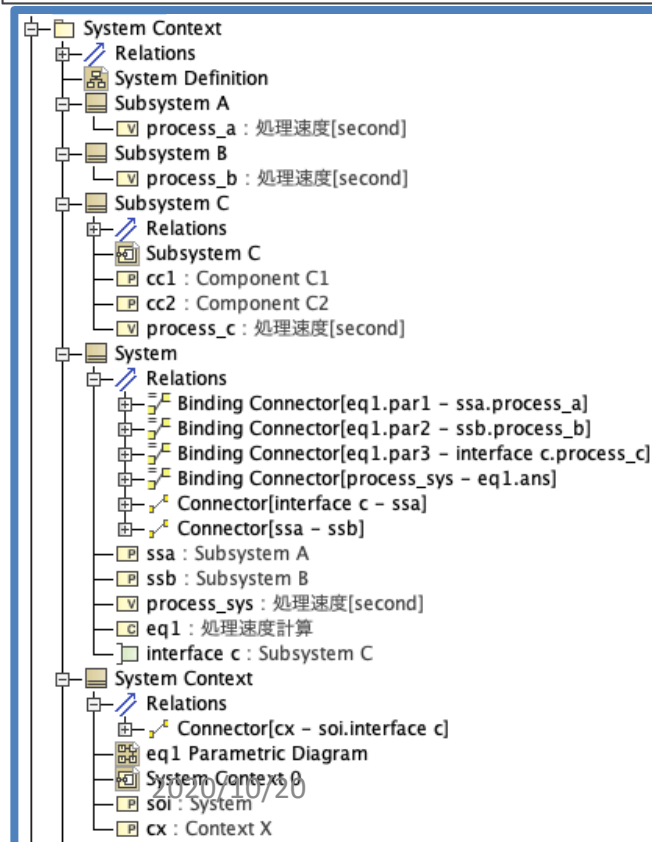
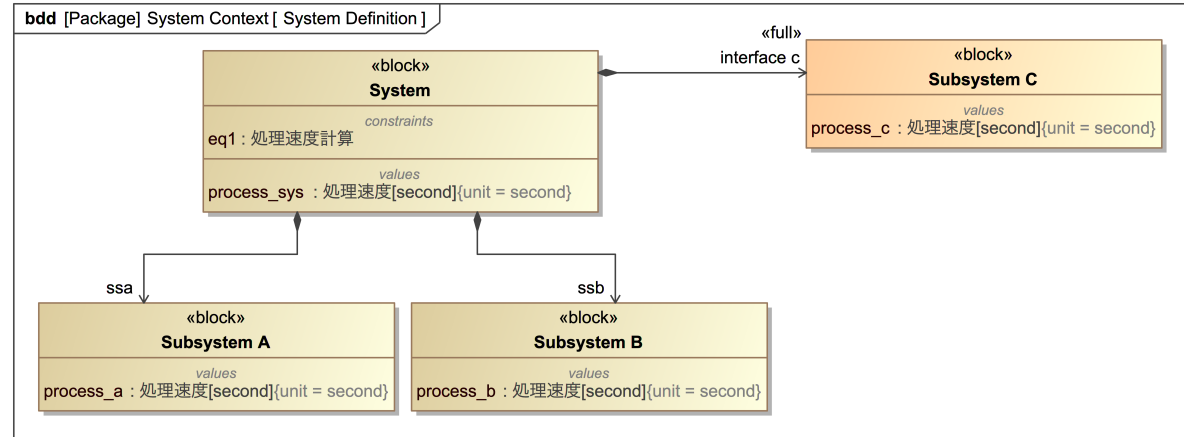
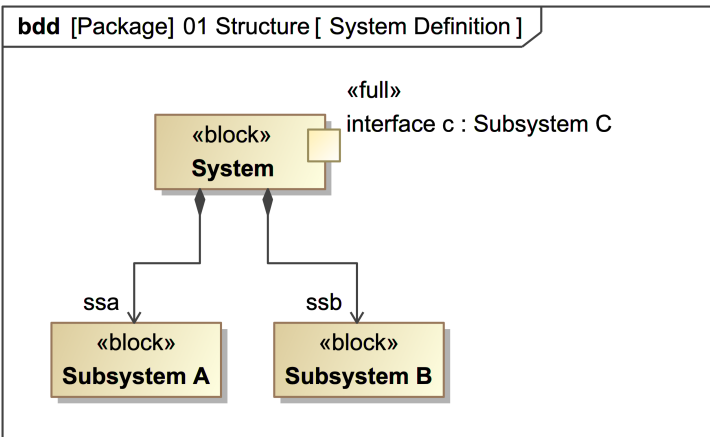


左図と全く同じモデル。
ツール上の表示方法を変えたのみ。



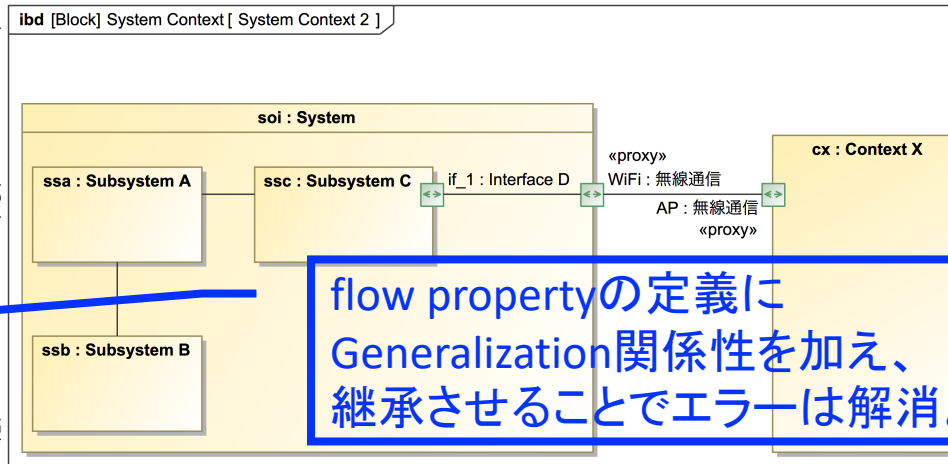
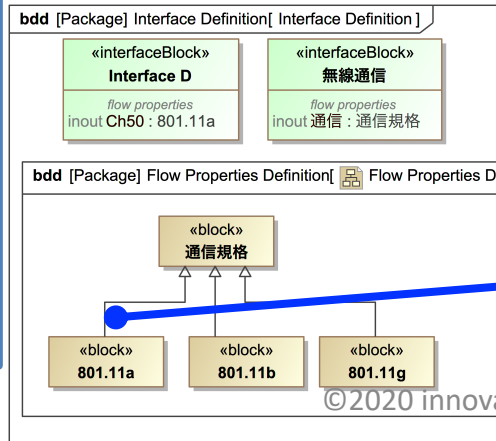
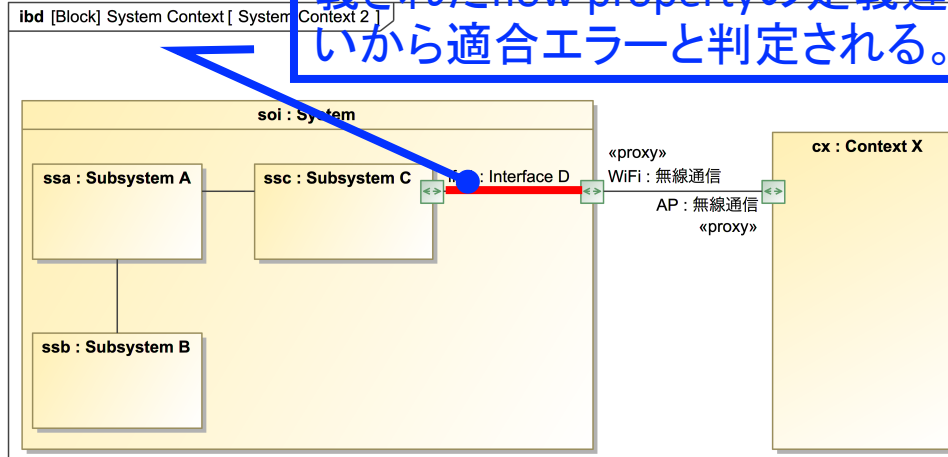
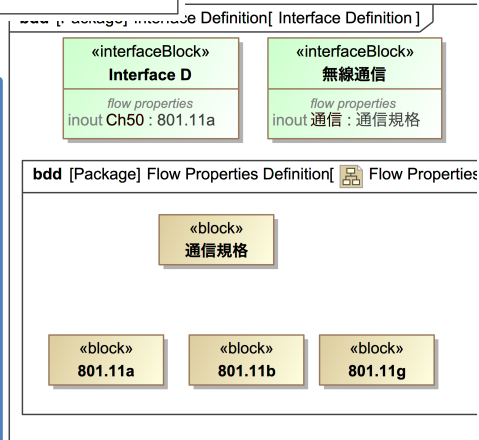
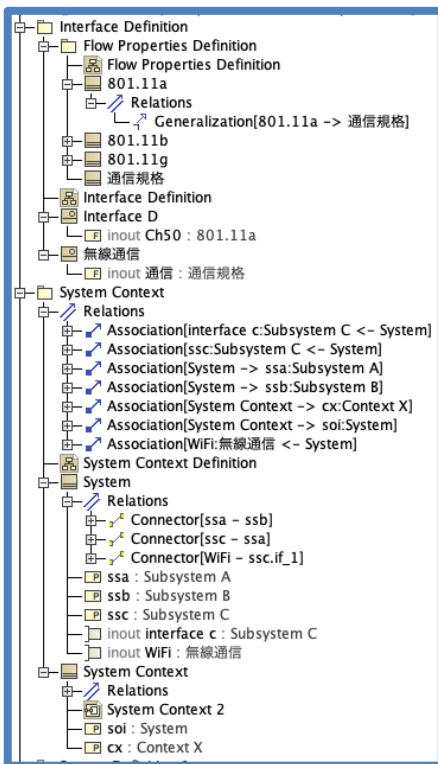
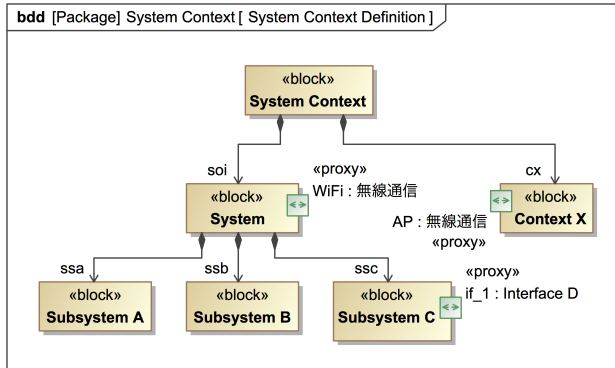
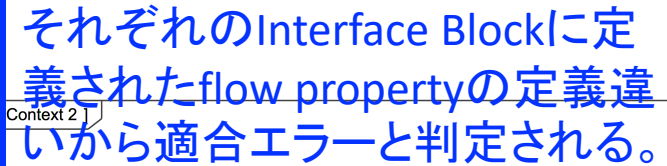
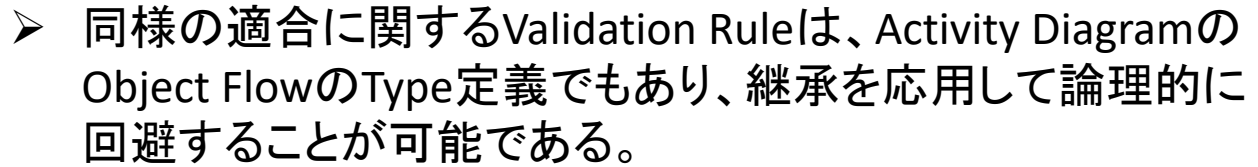
本記述では、cx, cc1/cc2, ssaというそれぞれ異なる階層同士が接続されていることに留意。

④【インタフェース記述方法】Full Portのvalue propertyを示し他のpartのvalue propertyと共にParametric Diagramを作成する記述方



Full Port内のvalue propertyを明示的に表示した状態。Part propertyと遜色ないことが分かる。

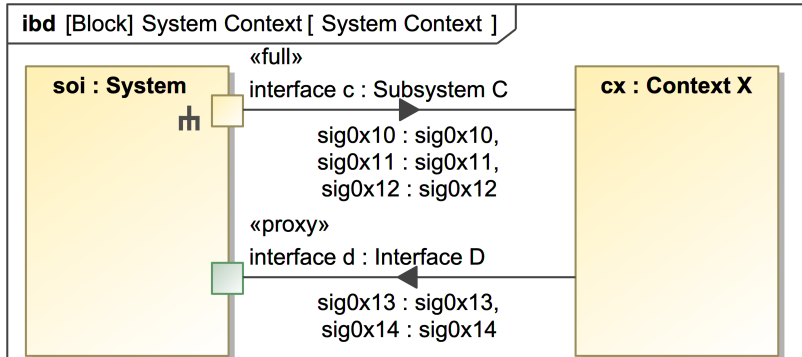
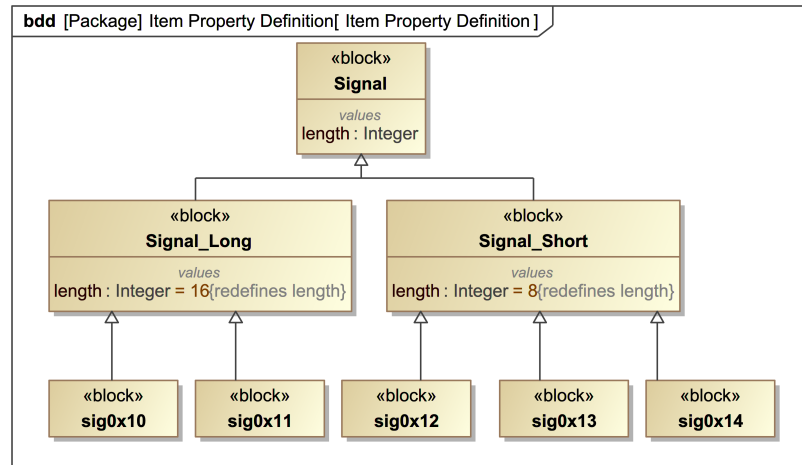
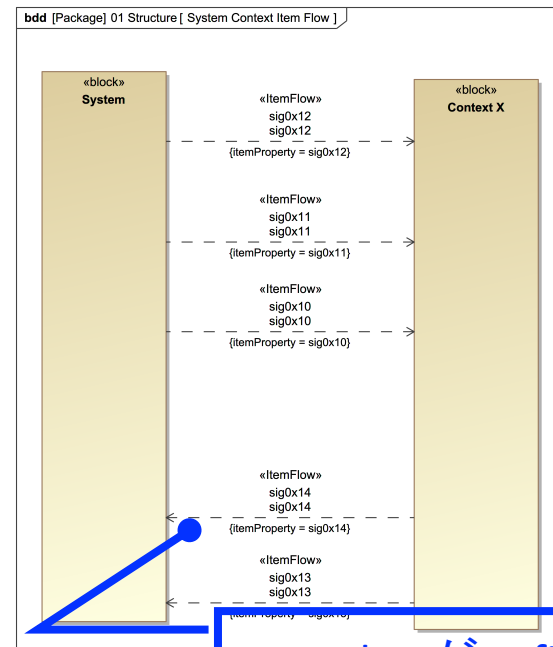
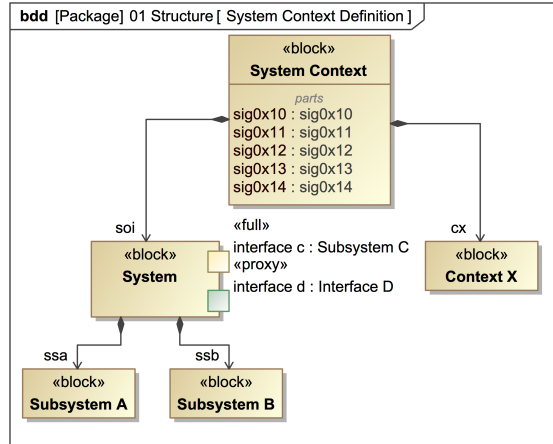
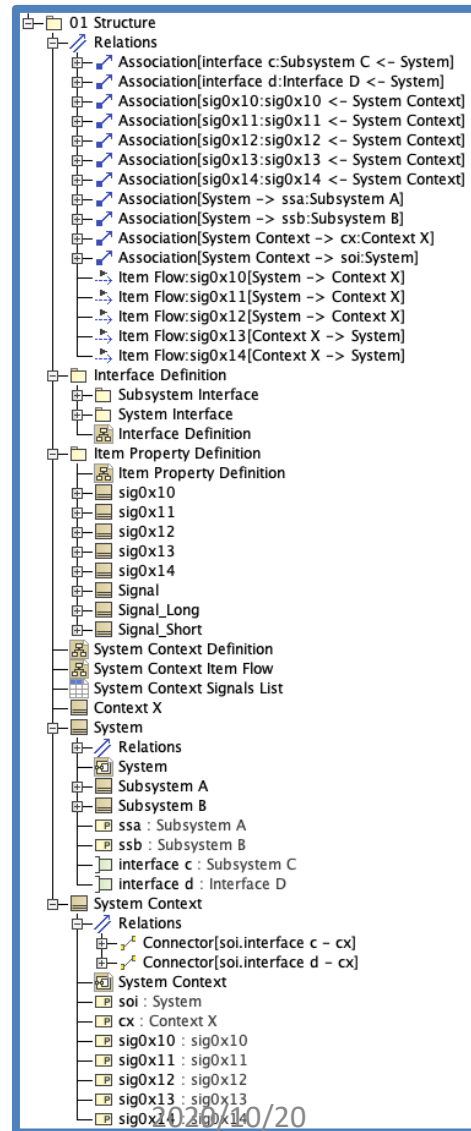
⑤【インタフェース記述方法】Proxy Portのflow property定義にGeneralizationによる継承を応用した記述方法



flow propertyの定義に
Generalization関係性を加え、
継承させることでエラーは解消。

⑥【インタフェース記述方法】Item Flowのitem propertyをpart propertyを用いて定義した記述方法

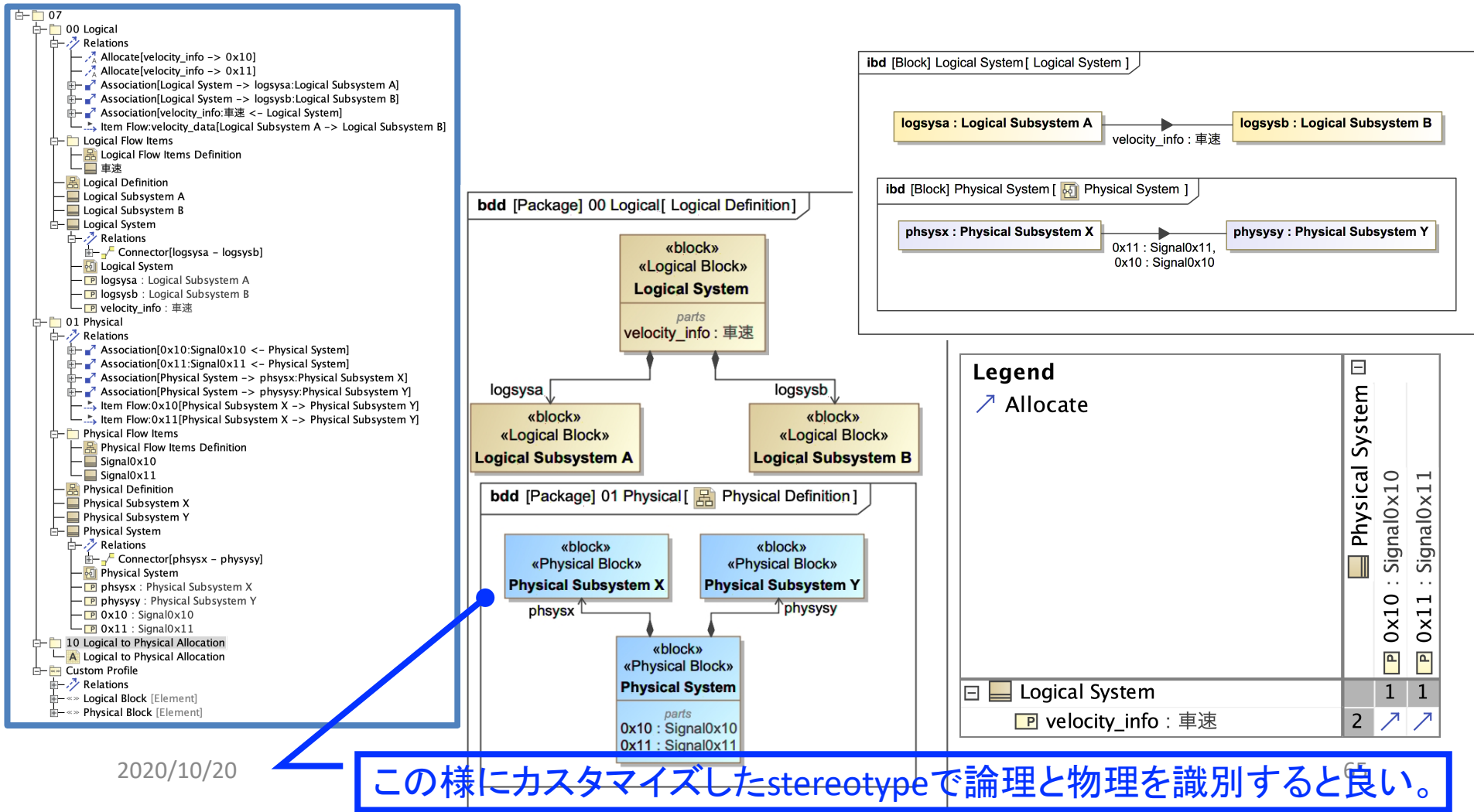
- System Context以下に、soi、cxと同列にsig0x11～sig0x14を管理することができる。すなわち、通信メッセージについても、機能や物理と同様に、具体的な要素として定義が可能となる



Item FlowがDefinition、item propertyがUsageである。

⑦【インタフェース記述方法】論理インタフェースと物理インタフェースを表現し分け、割り当て関係を明示するためのitem propertyのallocationの記述方法

- 通信や信号などの記述をさらに立体的にすることが可能であり、情報の活用自由度が高まることが期待できる。



2020/10/20

この様にカスタマイズしたstereotypeで論理と物理を識別すると良い。

SysMLを用いたシステム記述方法に関する情報

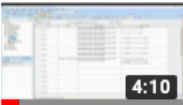
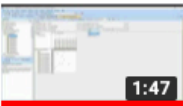
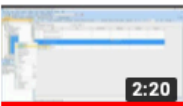
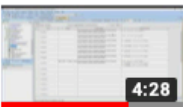
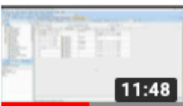
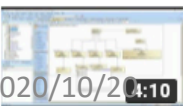
1. Ways to Approach Efficient Modeling, Booz Allen Hamilton
2. HYPERMODELING TUTORIAL, University of Detroit Mercy
3. Bespoke Modeling : Using Customization to Improve Model Quality and Modeler Productivity, SAIC

SysMLを用いたシステム記述方法に関する情報

1. Ways to Approach Efficient Modeling, Booz Allen Hamilton

https://www.youtube.com/playlist?list=PLY7P_TIXSex3DbXaT2DFSZW24byhCUXQk

基本的だが、非常に実践的なシステム記述方法の工夫が紹介されている。

- 1  **NMWS2016 DocumentingModelElements**
System Modeling
4:10
- 2  **NMWS2016 SmartPackages**
System Modeling
6:31
- 3  **NMWS2016 DependencyMatrices**
System Modeling
1:47
- 4  **NMWS2016 Inheritance**
System Modeling
2:20
- 5  **NMWS2016 MetaChainNavigation**
System Modeling
4:28
- 6  **NMWS2016 FlowsViaMetaChain**
System Modeling
11:48
- 7  **NMWS2016 Variants**
System Modeling
4:10

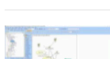
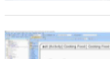
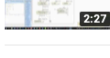
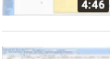
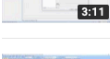
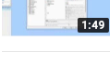
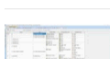
2020/10/20

SysMLを用いたシステム記述方法に関する情報

2. HYPERMODELING TUTORIAL, University of Detroit Mercy

https://www.youtube.com/playlist?list=PLDz4YEQgpXfzrPpUHVIRT8H67koH_FTfo

応用的な実践的システム記述方法の工夫が多く紹介されている。

1		Hypermodeling: Initial Release Systems Architecture Guild 0:23
2		Hypermodeling: Source Content Systems Architecture Guild 3:55
3		Hypermodeling: UseCase Modeling Systems Architecture Guild 9:19
4		Hypermodeling: UseCase Relationships Systems Architecture Guild 4:08
5		Hypermodeling: Decomposing UseCases with Activity Diagrams Systems Architecture Guild 1:57
6		Hypermodeling: Decomposing UseCases with Operations Systems Architecture Guild 6:50
7		Hypermodeling: UseCase Decomposition and Activity Diagram Logic Systems Architecture Guild 4:28
8		Hypermodeling: UseCase Decomposition and Cloning Activity Diagrams Systems Architecture Guild 6:15
9		Hypermodeling: System Context Traceability Systems Architecture Guild 3:29
10		Hypermodeling: Modifying Activity Diagrams Systems Architecture Guild 2:27
11		Hypermodeling: Logical Architecture Systems Architecture Guild 8:25
12		Hypermodeling: Additional Variant Specific Activity Diagrams Systems Architecture Guild 6:04
13		Hypermodeling: Additional Variant Specific Activity Diagrams Systems Architecture Guild 5:34
14		Hypermodeling: Dissociation Systems Architecture Guild 3:40
15		Hypermodeling: Logical Internal Block Diagram Systems Architecture Guild 4:36
16		Hypermodeling: Part Property Multiplicities Systems Architecture Guild 0:55
17		Hypermodeling: Owning Interface Blocks Systems Architecture Guild 2:04
18		Hypermodeling: Assigning Events to Ports Systems Architecture Guild 2:35
19		Hypermodeling: Sequence Diagrams Systems Architecture Guild 1:47
20		Hypermodeling: Integrating IBDs Messages and Object Flows Systems Architecture Guild 4:46
21		Hypermodeling: Item Flow Additional Example Systems Architecture Guild 3:11
22		Hypermodeling: Item Flow Error Checking Systems Architecture Guild 2:18
23		Hypermodeling: Cloning Behaviors and Item Flow Realization Systems Architecture Guild 1:49
24		Hypermodeling: Physical Realization Systems Architecture Guild 5:05
25		Hypermodeling: Error Finding With Excludes Systems Architecture Guild 5:15
26		Hypermodeling: Metrics Suites Systems Architecture Guild 4:34
27		Hypermodeling: IBDs Systems Architecture Guild 1:43
28		Hypermodeling: Layout Templates Systems Architecture Guild 1:08
29		Hypermodeling: Creating Power Scenarios Systems Architecture Guild 5:55
30		Hypermodeling: Trapped Parameters Systems Architecture Guild 2:06
31		Hypermodeling: Cable Example Systems Architecture Guild 1:58
32		Hypermodeling: Threaded Fasteners Systems Architecture Guild 1:48
33		Hypermodeling: Validation Suites Systems Architecture Guild 6:42

2020/10/30

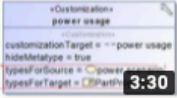
SysMLを用いたシステム記述方法に関する情報

3. Bespoke Modeling : Using Customization to Improve Model Quality and Modeler Productivity, SAIC

<https://www.youtube.com/playlist?list=PLDz4YEQgpXfy8sFLZrImBUF0VLdSNxyeF>

応用的なシステム記述方法の工夫が紹介されている。

- 1



MCES 2019 Power Scenario
Systems Architecture Guild
3:30
- 2



MCES 2019 Power Scenario Detail
Systems Architecture Guild
5:27
- 3



MCES 2019 Mass Rollup
Systems Architecture Guild
8:14
- 4



MCES 2019 Quality Checks
Systems Architecture Guild
5:32
- 5



MCES 2019 PreCommit Validation
Systems Architecture Guild
4:43
- 6

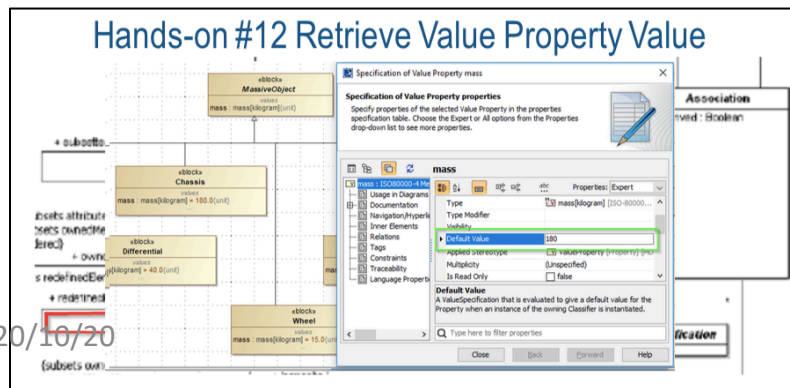
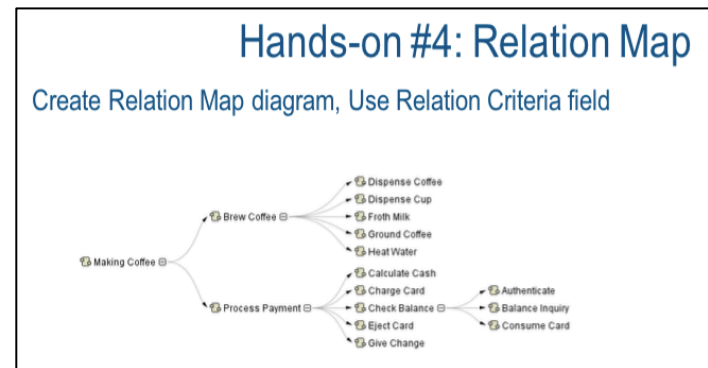


MCES 2019 Validation Driven Metrics
Systems Architecture Guild
2:28

Structured Expression Seminar, Dassault Systems

Legendの設定、High Order Operationの活用など、汎用性の高い内容が多い。

- [illegible]



- 2020/10/20