FMI 活用ガイド

Ver.2.0.1

2024 年 6 月 1 日

Revision	日付	主な内容
1.0.0	2018/10/01	Ver1.0.0 リリース
1.0.1	2018/11/01	6 章：6.2.4 を追加、6.2.5 の最少計算時間の誤記を修正
2.0.0	2023/10/31	Ver2.0.0 リリース
2.0.1	2024/6/1	Ver2.0.1 リリース

目次

本ガイド作成にあたり、自動車技術会 自動車制御とモデル部門委員会 FMI 活用・展開 WG に参加、協力頂いた個人、企業・団体は以下の通りです。

江嶋 睦仁	株式会社 IDAJ
市原 純一	AZAPA 株式会社
守屋 一成	AZAPA 株式会社
関末 崇行	アンシス・ジャパン 株式会社
岩ヶ谷 崇	サイバネットシステム株式会社
緒方 洋介	シーメンス 株式会社
VIRY、 Guillaume	ダッソー・システムズ 株式会社
都築 勝也	dSPACE Japan 株式会社
吉松 俊	dSPACE Japan 株式会社
内田 裕美	株式会社 電通総研
上山 慶一	株式会社 デンソー
三輪 修也	株式会社 デンソー
荒木 大	東芝デジタルソリューションズ株式会社
平野 豊	HIRANO Research Lab.
猿木 恭文	HEXAGON 株式会社
平井 信之	HEXAGON 株式会社
斉藤 春樹	日産自動車 株式会社
神野 英章	株式会社 本田技術研究所
平尾 俊一	株式会社 本田技術研究所
赤阪 大介	The MathWorks GK
宅島 章夫	The MathWorks GK
遠藤 駿	マツダ 株式会社
小森 賢	マツダ 株式会社
武内 宏明	マツダ 株式会社
佐藤 裕司	三菱スペース・ソフトウエア 株式会社
GAO、 Rui	モデロン 株式会社

（企業・団体名で 50 音順）

また、本ガイド作成にあたり、下記の、個人、企業・団体の皆様にもご協力をいただきました。

石川 誠司	イータス株式会社
渋谷 賢佑	名古屋大学
重松 浩一	名古屋大学
船橋 豊	ルネサスエレクトロニクス株式会社

3V-SG（仮想的手法を用いた制御検証研究会）

（企業・団体名で 50 音順）

第 1 章　本ガイドについて

1.1.　本ガイドの目的

本ガイドは自動車システムの開発においてシミュレーションを活用しているエンジニアを対象としたシミュレーションモデルの交換および接続のための手引き書です。

自動車システムは高機能と高性能が求められ、その複雑なシステムの開発においてシミュレーションを活用したモデルベース開発の重要性が増しています。そして、複数のシステムや部品を組み合わせてシステム全体の振る舞いを検証するために、社内の部署間あるいは他社とシミュレーションモデルの流通が求められています。

本ガイドではシミュレーションモデルを交換および接続するための基礎的な技術や注意点を解説しています。

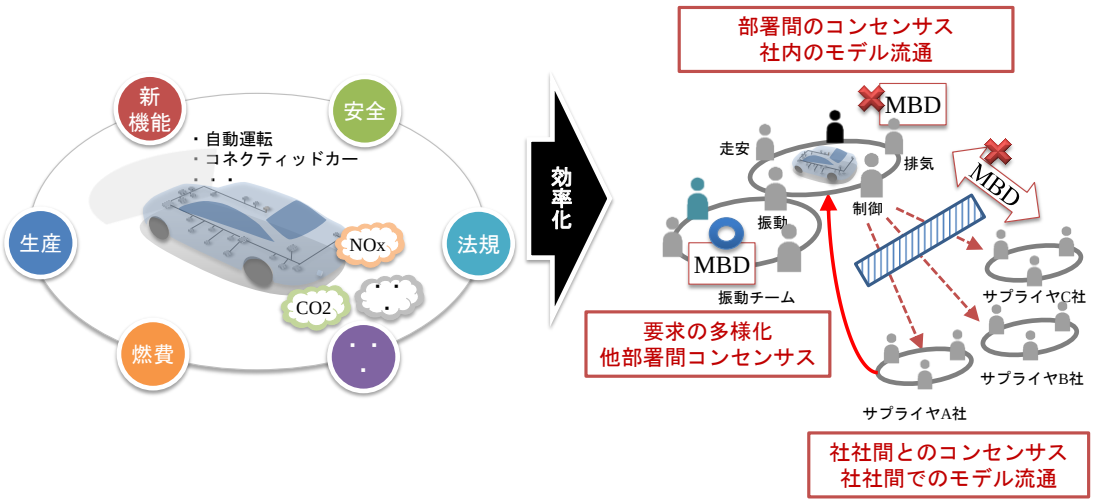


図 1.1.1 図 1.1 モデル流通のイメージ

（自技会　振動騒音シンポジウム（2017）、「FMI によるモデル流通を目指して」より引用）[1]

1.2. はじめに

自動車システムの開発においてシミュレーションは設計／試作／テストの一連の全ての工程で活用されています。コンピュータの性能向上とソフトウェアの進歩に伴い、シミュレーションの活用が進んでいます。モデルを用いたシミュレーション環境の構成要素は下記です。

表 1.2-1　シミュレーション環境の構成要素

モデル	シミュレーション対象の物理動作やコントローラ動作を 下記により抽象化したもの ・グラフィカルな記述 ・モデリング言語やプログラミング言語 ・数式記述、表、マップなど
ソルバ	積分等の数値演算を行うための機能
アプリケーションソフトウェア	シミュレーションツールの動作プログラム
オペレーティングシステム	コンピュータの基本プログラム
コンピュータ	プログラムを実行する物理装置、ネットワーク通信環境も含む

1.3.　背景

1.3.1.　モデル流通への期待

自動車システムの開発は、まずシステムの構成要素となる部品の設計／試作を行い、次にそれらを組み合わせた試作車を用いてテストが行われてきました。しかし、高機能化と高性能化による複雑化と開発期間短縮への要望に応えられなくなり、従来型の開発スタイルでは限界が見えてきました。そこで、仕様書と実部品という関係から、仕様書とそれに対応するシミュレーションモデルへの置き換えが進み、システムの構成要素となるサブモデルを組み合わせた試作車モデルを用いたシミュレーションでテストを行うモデルベース開発へ変わりつつあります。そして、社内の部署間あるいは他社とのシミュレーションモデルの流通が求められています。

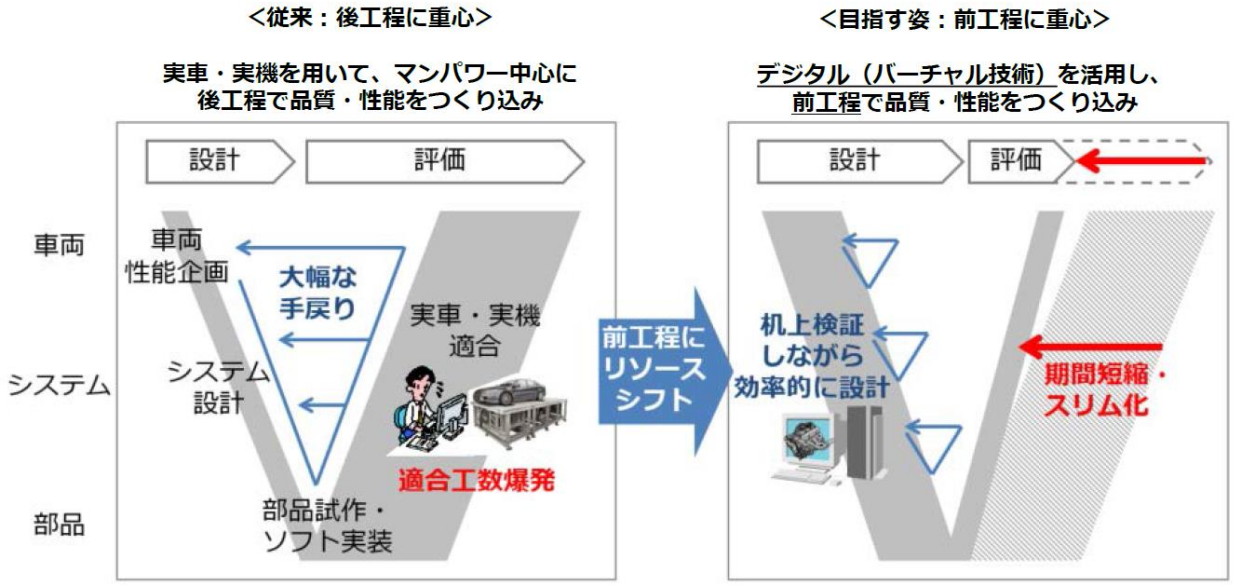


図 1.3.1　「車」の作り方の革新

（経済産業省　自動車新時代戦略会議（第１回）配布資料より引用）[2]

1.3.2.　モデル流通の課題

シミュレーションモデルの記述には、様々なシミュレーションツールが使用されているため、企業内の部門間あるいは企業間でモデルを流通するには、相互のシミュレーションツールの統一か、異なるシミュレーションツールにより作成されたモデルの接続が必要です。特に、自動車システム開発に関係する全ての企業でシミュレーションツールを統一するのは現実的でないため、異なるシミュレーションツール間でのモデルの交換および接続するためのインターフェースの共通化が課題となっています。

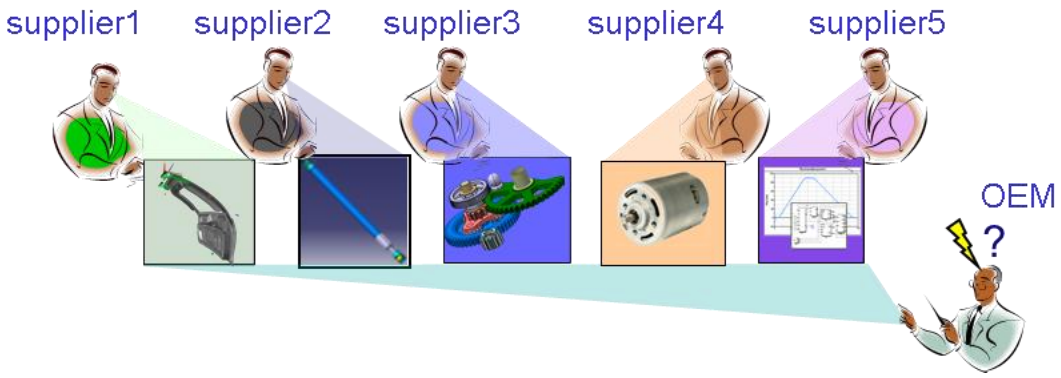


図 1.3.2　OEM とサプライヤ間モデル流通

（8th International Modelica Conference 2011 前刷り資料より引用）[3]

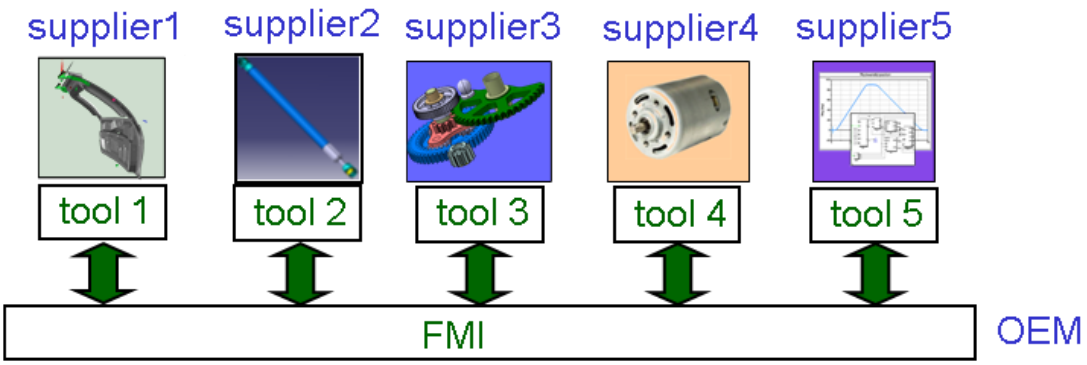


図 1.3.3　FMI による OEM とサプライヤ間モデル流通

（8th International Modelica Conference 2011 前刷り資料より引用）[3]

1.3.3.　モデル流通の標準化

シミュレーションツールに依存しないモデル接続のための共通インターフェースとして、欧州の公的プロジェクトが規格化した FMI（Functional Mock-up Interface）が世界的に普及しています。欧州の自動車業界におけるシステム開発手法の効率化を推進する非営利団体 ProSTEP iViP Association が策定したガイドラインでも、FMI は、モデル接続・交換のための標準インターフェースの一つとして推奨されています。自動車技術会でも 2012 年 3 月に「国際標準記述によるモデル開発・流通検討委員会」を設置し、モデル接続技術検討 WG において FMI によるモデル接続検証と、モデルの交換および接続の推進が行われてきました。その後、この活動は「自動車制御とモデル部門委員会」の FMI 活用・展開検討 WG に引き継がれ、モデリングツールに依存せずに異なるシミュレーションツール間でモデルの交換および接続する一般的な手法を検討しています。FMI 活用・展開検討 WG では、以下の目的で活動を継続しています。

① FMI によるモデル接続手法の検証
② 技術課題の洗い出しと対応法の検討
③ 得られた知見の展開（学術講演会・フォーラムの開催、活用ガイドの出版）
④ Modelica Association（FMI 規格の策定・推進非営利団体）や関連団体との連携

その成果をまとめ、2018 年に、「FMI 活用ガイド Ver.1.0.1」が発表され、自由にダウンロードできるようになっています。今回、2018 年以降の FMI 規格の更新や拡張の結果を踏まえ、「FMI 活用ガイド」を Ver. 2.0 として更新しました。本ガイドが、自動車業界でツールによらないモデル接続・流通が活発となり、モデルベース開発が推進されることに貢献できることを切に望みます。

第 2 章　FMI の基本

この章では、FMI に関する基本的な知識や情報について説明します。

2.1 は、FMI の成り立ちについて説明します。

2.2 では、FMI の 3 つの動作モードである Model Exchange（ME）、Co-Simulation（CS）と Scheduled-Execution（SE）の基本構造について説明します。2.3、2.4 及び 2.5 では、それぞれの構成と特徴について解説します。

2.6 では、FMI 形式でモデルを受け渡す際の互換性に関する注意点を説明します。

2.7 では、FMI 形式でモデルを受け渡す際の情報隠蔽に関する注意点を説明します。

2.8 では、FMI 規格をサポートするツールの探し方と調べ方を説明します。

2.1.　FMI の成り立ち

2008 年～2011 年の間、EU プロジェクト Information Technology for European Advancement（ITEA2）の一環として、MODELISAR というプロジェクトが実施されました。その結果、以下の要件を満たすモデル接続のための共通インターフェース規格 FMI（Functional Mock-up Interface）ver. 1.0（以下では FMI1.0 と記します）が策定されました。FMI1.0 では Model Exchange（ME）、Co-Simulation（CS）の 2 種類の動作モードが規定されましたが、この二つの規格は別々に作られました。

MODELISAR プロジェクトの終了後、FMI の活動は，モデルベース開発に関わる企業関係者や大学研究者、ツールベンダの技術者などからなる非営利団体 Modelica Association（以下では MA と記します）に引き継がれました。MA において、FMI1.0 で不足していた機能の追加や、仕様の曖昧性の見直しが行われ、2014 年 7 月に FMI for Model Exchange and Co-Simulation Ver. 2.0（以下では FMI2.0 と記します）が制定されました。FMI2.0 では、Model Exchange（ME）、Co-Simulation（CS）の 2 種類の動作モードが一つの規格として統合されました。その後、FMI2.0 は小改訂が行われており、現在の FMI2.0 の最新規格は 2022 年 11 月に発行された Ver.2.0.4 です。

FMI1.0 および FMI2.0 は、モデル流通の業界標準として，現在までに 200 に近いツールでサポートされています。

この一方で、高度な協調シミュレーションアルゴリズム、V-ECU（仮想電子制御ユニット）のパッケージ化やシミュレーションなどの新しいニーズとユースケースが発生しており、MA の中でこれらを可能にする新機能が検討され、2022 年 5 月に、FMI Ver.3.0（以下では FMI3.0 と記します）の規格が制定されました。FMI3.0 では、3 種類目の動作モードとして Scheduled Execution（SE）が追加されたほかに、データ型の拡充、クロック信号やイベント駆動などのさまざまな新機能を持ちます。

本書では最新の FMI3.0 についても解説しますが、FMI3.0 をサポートするツールはまだ少数であり、FMI3.0 を利用できる機会は多くはありません。そこで本書の説明の多くは FMI2.0 を中心に取り扱います。

2.2.　FMI の動作モードと FMU ファイルの構造

2.2.1.　FMI の動作モード

FMI 規格では、FMU（Functional Mock-up Unit）と呼ぶモデル部品インターフェースを使って異種ツール間でモデルを受け渡します。FMI には、Model Exchange (ME)、Co-Simulation (CS)、Scheduled Execution（SE）の 3 つの動作モードがあります。

ME と CS は FMI 1.0 と FMI 2.0 からある動作モードです。SE は FMI3.0 で新たに導入された動作モードです。

ME の場合は、計算モデルだけを FMU にエクスポートして受けわたして、FMU をインポートするツール側のソルバを使用します。CS の場合は、計算モデルとソルバをセットで FMU にエクスポートして異種ツール間でモデルを受け渡しますので､FMU をインポートするツール側で FMU を計算するときは、FMU の内部に組み込まれたソルバを使用します。ここが、ME と CS の大きな違いです。

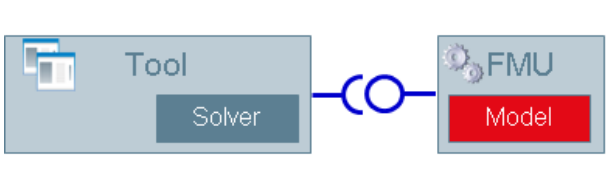


図 2.2.1(a)　FMU にソルバを持たない ME

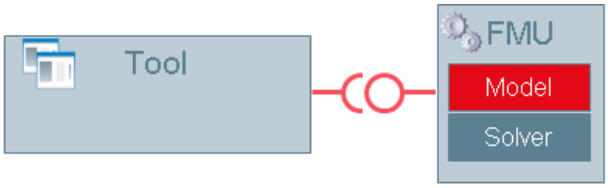


図 2.2.1(b)　FMU にソルバを内包する CS

（8th International Modelica Conference 2011 講演資料より引用）[3]

SE は、V-ECU（仮想電子制御ユニット）のパッケージ化とシミュレーションが想定用途で、ME や CS とは構造上の違いがあります。モデルパーティションとよぶ、制御コントローラのタスクコードに相当する部分をモデルとして受け渡して、インポートするツール側に用意されたスケジューラを使ってモデルを動かす仕組みになっています。

2.2.2.　FMU のファイル構造

FMU の拡張子は.fmu であり、その実態は ZIP 形式で圧縮されたものです。概念を図 2.2.2 に示します。FMU は、Windows OS における実行形式の dll や Linux OS における実行形式の so とモデルの詳細説明である xml 形式の model Description File からなりたっています。(実行形式モジュールの説明追加)。ソースファイル隠蔽の場合⇒「実行形式の dll を生成するためにはソースファイルが必要です。FMU を生成するツールでは、dll 生成後にソースファイルを削除し、隠蔽可能としています。この設定方法は、ツールに依存しますので、使用するツールのマニュアルを参考にして下さい。」

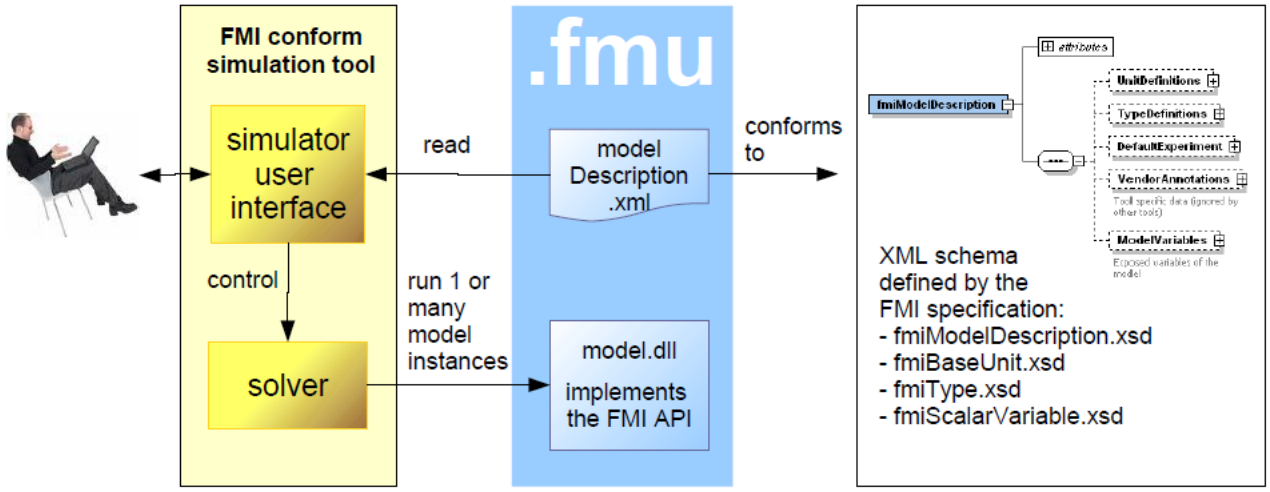


図 2.2.2　FMU の構造

（8th International Modelica Conference 2011 前刷り資料より引用）[5]

model Description File の構造を図 2.2.3 に示します。先に述べたように、FMI1.0 では ME と CS は企画段階から別々に立案・策定され、共通の構造がないので、FMI2.0 の構造を用いて説明します。

なお、FMI 1.0 と 2.0 との間には、互換性は保たれていないので取り扱いには注意が必要です。

はじめに、ME か CS かの区別の記述があり、その後に実装、単位、変数、属性、構造とその属性などが細かく記述されています。

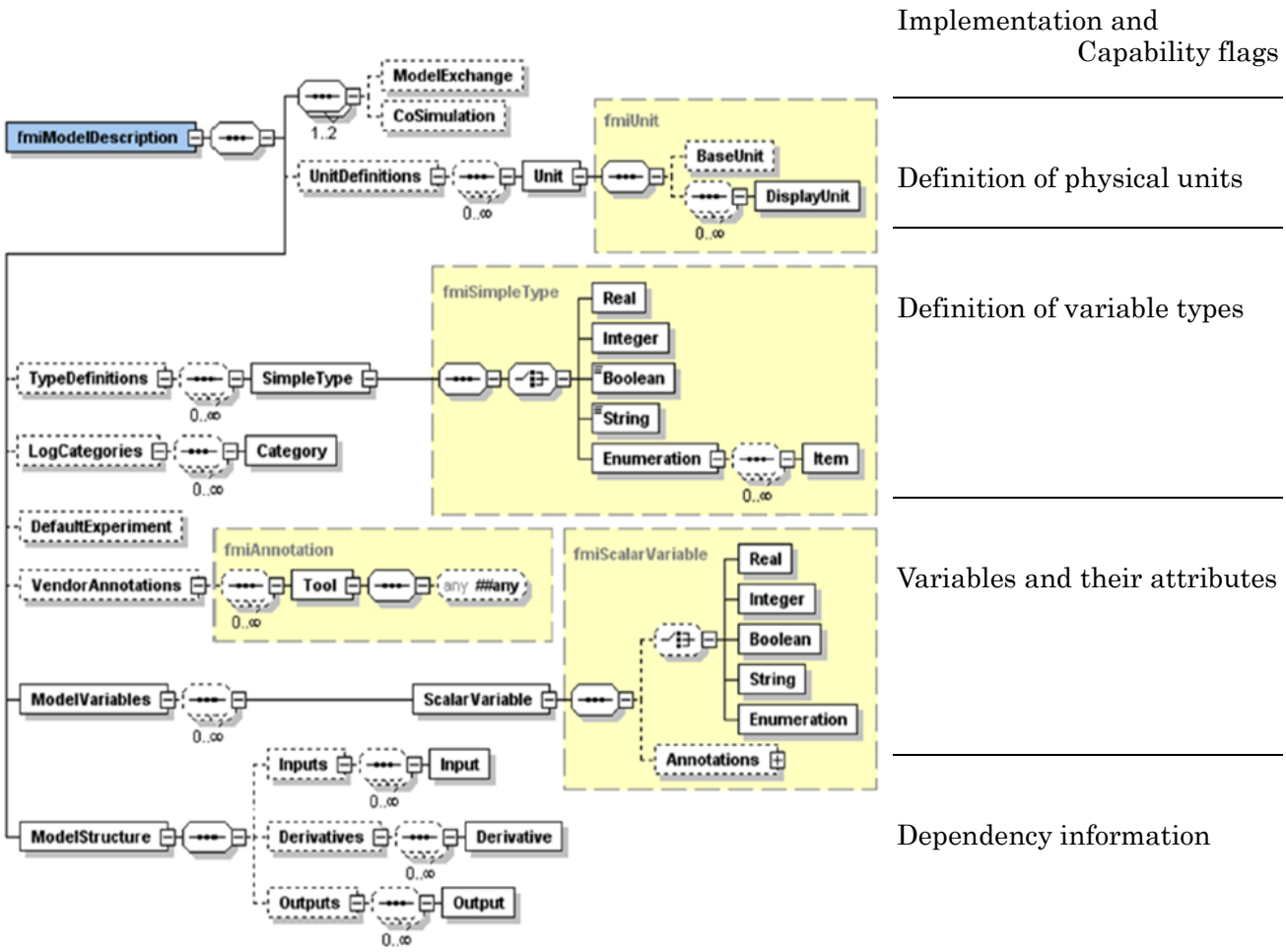


図 2.2.3　model Description File の構造

（2nd Japanese Modelica Conference 基調講演資料より引用）[6]

2.3.　Model Exchange（ME）の構成と特徴

2.3.1.　1ME における信号の流れ

ME モードでは FMU を取り込んだシミュレーションツールのソルバを用いて FMU 内の計算は行われます。従って、FMU とその周辺のモデルの時間遷移が同一のソルバで同時に実施されるため、時間同期が保証されます。

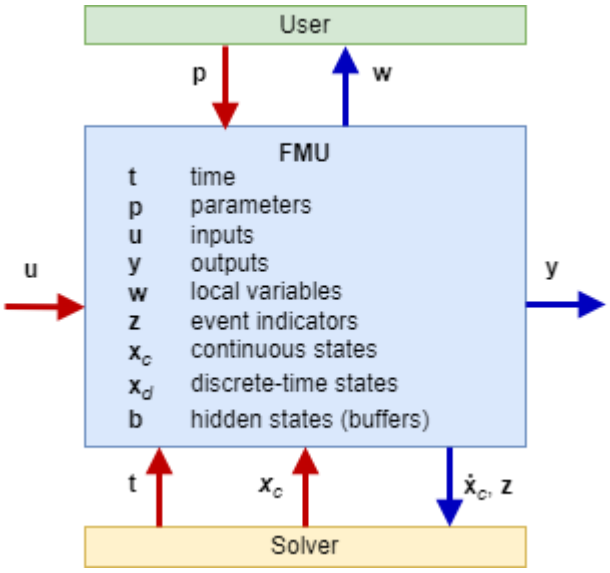


図 2.3.1　ME モードの信号の流れ

（FMI 仕様書：Functional Mock-up Interface Specification_Version 3.0 より引用）[7]

2.3.2. ME でのモデル接続例

図 2.3.2(a)に非因果系ツールに取り込み Model Exchange で接続した FMU と信号の流れの例を示します。このモデルは 2 つの FMU から構成されます。図 2.3.2(b)は FMU にする前のモデルで、赤色破線が図 2.3.2(a)のそれぞれの FMU に相当します。図 2.3.2(c)に構造の概念図を示します。図 2.3.1 に示すように FMU は取り込んだツール上のソルバでシミュレーションを実施します。

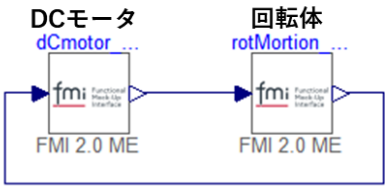


図 2.3.2 (a)　Model Exchange モードでの接続例(FMU)

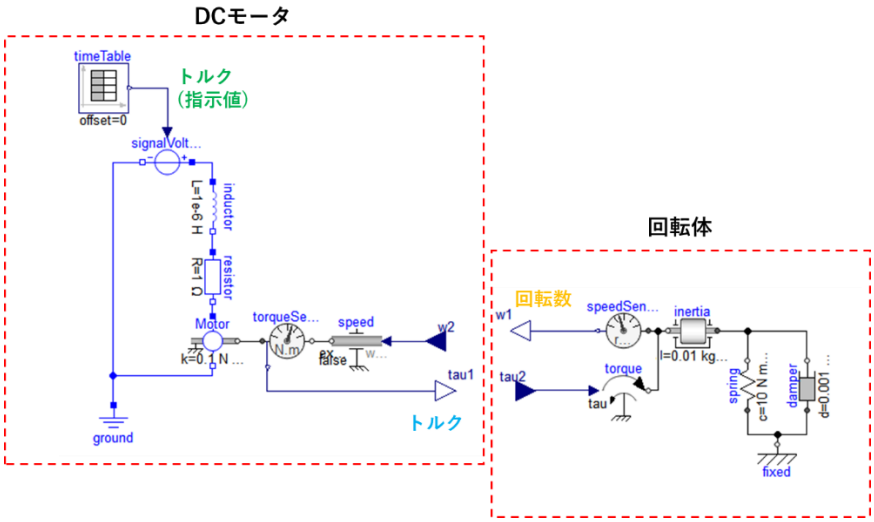


図 2.3.2(b)　Model Exchange モードでの接続例(元モデル)

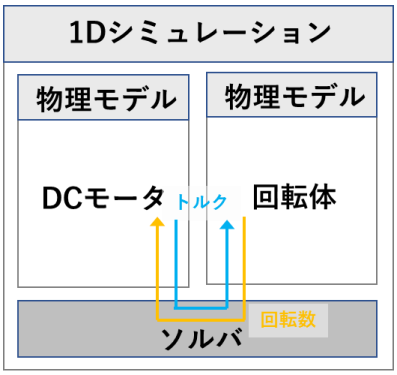


図 2.3.2(c)　ME モデル（図 2.3.2(b)）の補足

2.4. Co-Simulation（CS）の構成と特徴

2.4.1. CS の 3 つの形態

CS は、運用により 3 つの形態に分類されます。

もっとも基本的なものが Stand Alone（図 2.4.1）であり、PC 上でマスタとなるツールと FMU が 1 つのコアの上で動作します。

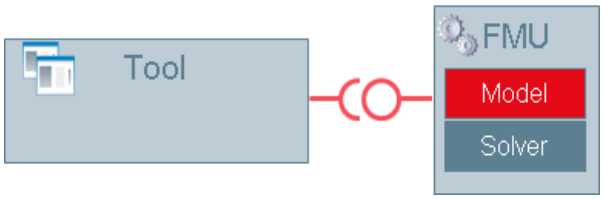


図 2.4.1 Stand Alone Co-Simulation

（8th International Modelica Conference 2011 講演資料より引用）[3]

2 つ目の形態は、Tool Coupling（図 2.4.2）であり、マスタとなるツールと FMU とが別々のプロセッサもしくは別々のコアの上で動作します。FMI Wrapper については、Functional Mock-up Interface Specification_Version 3.0 [7] を参照して下さい。

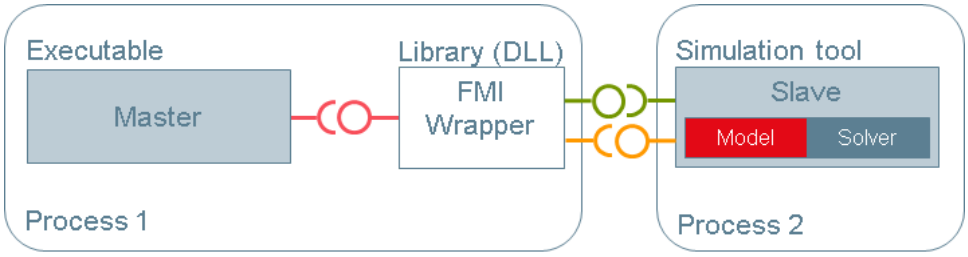


図 2.4.2 Tool Coupling Co-Simulation

（8th International Modelica Conference 2011 講演資料より引用）[3]

3 つ目の形態は、Distributed（図 2.4.3）で、莫大なコンピュータリソースを必要とするような大規模システムで適用されます。クライアント・サーバ方式の基本的な分散環境の上に、FMI 規格に準拠した構成になっています。

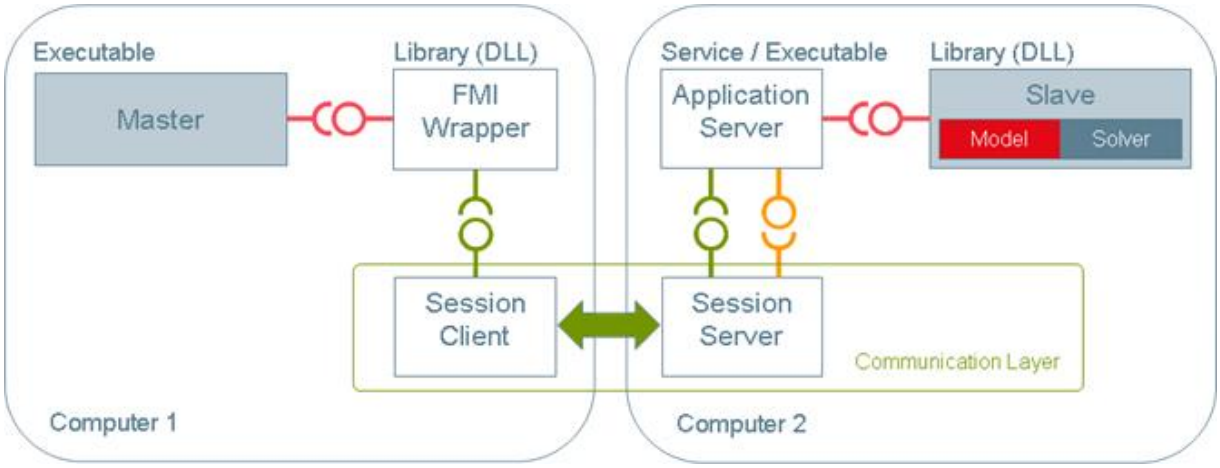


図 2.4.3 Distributed Co-Simulation

（8th International Modelica Conference 2011 講演資料より引用）[3]

2.4.2. CS における信号の流れ

CS における信号の流れを図 2.4.4 に示します。(u)、(y)が FMU の入出力であり、FMU 内部のソルバによって計算されます。この入力(u)や出力(y)は、第 3 章にて後述する Communication Step Size で設定された時間ごとに FMU を取り込んだツールとの間で信号授受を行います。

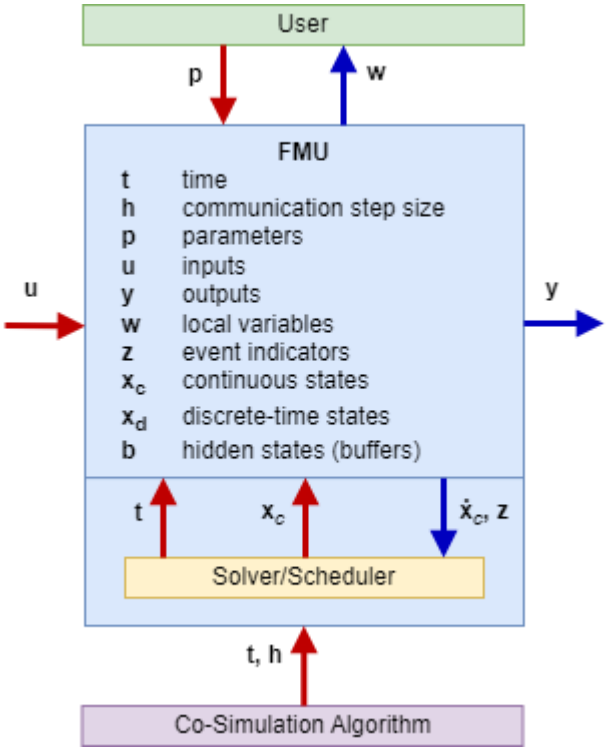


図 2.4.4　CS モードの信号流れ

（FMI 仕様書：Functional Mock-up Interface Specification_Version 3.0 より引用）[7]

2.4.3. CS でのモデル接続例

図 2.4.5(a)に非因果系ツールに取り込み Co-Simulation で接続した FMU と信号の流れの例を示します。このモデルは 3 つの FMU から構成されます。図 2.4.5(b)は FMU にする前のモデルで、赤色破線が図 2.4.5(a)のそれぞれの FMU に相当します。この図では各信号が直接 FMU 間で授受されているように見えますが、実際は図 2.4.5(c)に示すように、取り込み側を介して信号授受を行います。各 FMU は独自のソルバを持ち、それぞれ計算を行い、Communication Step Size 毎に取り込み側とこのソルバを介して信号授受することにより、全体のシミュレーションを実行します。

シミュレーションツール上で取り込んだ各 FMU のプロパティを見ると、FMU ごとに Communication Step Size など通信に関するパラメータを設定する事ができます。

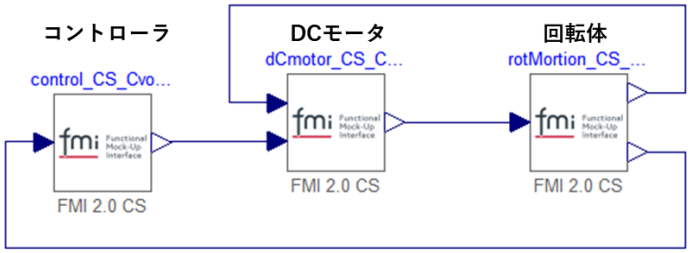


図 2.4.5(a)　Co-Simulation モードの接続例(FMU)

具体的には図 2.4.5(a)の DC モータに対する信号の送受信は、図 2.4.5(b)に示すようにコントローラからの電源へのトルクの指示値が FMU の取り込み側を介して DC モータに送信されます。指示値を受けて DC モータがトルクを出し、それから算出された回転体からの回転数と角度がまたそれぞれ取り込み側を介してコントローラにフィードバックされます。

この受け渡しのタイミングは、Communication Step Size（CSS）で設定された時間間隔になります。コントローラからのトルク指示値は、取り込み側を介していますので、DC モータが取り込み側から受け取れるのは次の CSS のタイミングになります。これを時間軸で表現すると図 2.4.5(d)のようになります。

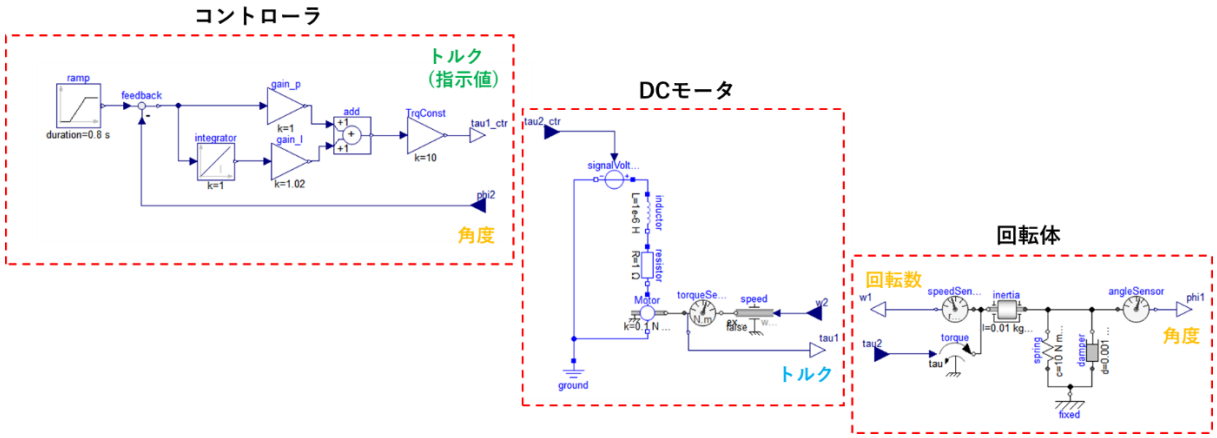


図 2.4.5(b)　Co-Simulation モードの接続例(元モデル)

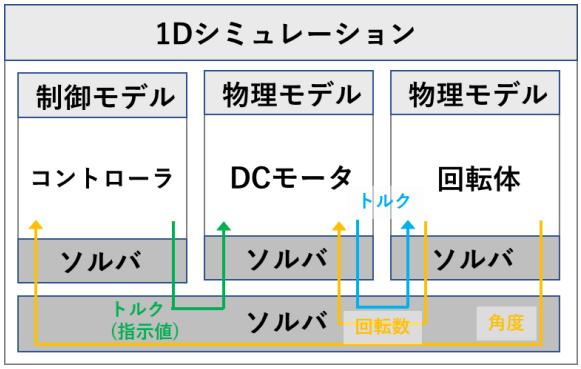


図 2.4.5(c)　CS モデル（図 2.4.5(b)）の補足

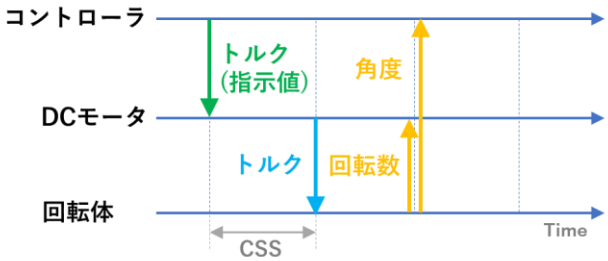


図 2.4.5(d)　CS モデル（図 2.4.5(a)~(c)）の補足

2.5. Scheduled Execution (SE) の構成と特徴

2.5.1. SE モードの FMU の構成

Scheduled Execution（SE）インターフェースは FMI3.0 で新たに導入された動作モードで、V-ECU（仮想電子制御ユニット）のシミュレーションに使用することを想定しています。

SE の FMU は図 2.5.1 に示すように、複数のモデルパーティションにより構成することができます。モデルパーティションとは、制御コントローラのタスクに対応するアルゴリズムと、アルゴリズムの中で使用する変数を意味します。個々のモデルパーティションに対して、駆動用のクロック（t）を関連付けることができ、FMU の外部のスケジューラからクロックを使ってモデルパーティションを動かします。マルチプログラム方式（あるいはマルチタスク、マルチプロセス方式）で、複数のプログラムを見かけ上、同時に実行するときに、どのような順番で、プログラムを実行するかなどを決定し、実行する仕組みがスケジューラです。通常は、OS のカーネル部分にある主要な機能の 1 つですが、SE の FMU を駆動するためには、このようなスケジューラが必要です。

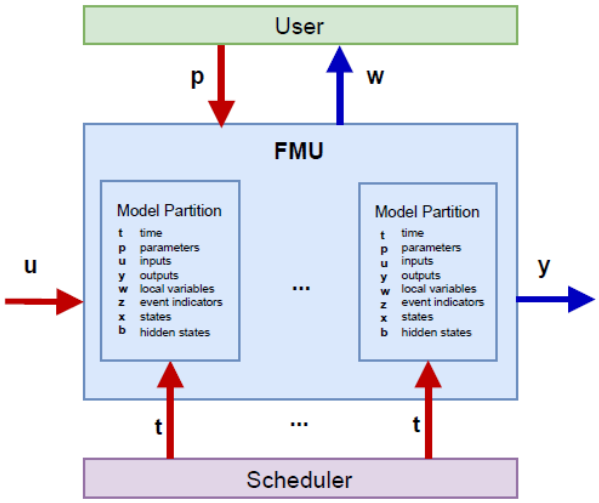


図 2.5.1　SE モードの信号の流れ

（FMI 仕様書：Functional Mock-up Interface Specification_Version 3.0 より引用）[7]

このように、SE の FMU の中のモデルパーティションは、外部のスケジューラからいつでもアクティベートが可能です。

クロックは、FMI3.0 において新しく導入された仕様です。クロックを用いて、モデルパーティションの実行やイベントのタイミングを制御することができます。クロックの発生タイミングは、固定周期駆動、可変周期駆動，あるいは，突発的なタイミングのイベント等による駆動があります。SE の FMU から FMU をインポートしたツール側に対して、どのようなタイミングでモデルパーティションをアクティベートすることができるかの情報を提供します.

モデルパーティションから他のモデルパーティションの実行を制御する方法として、クロックのカウントダウンを用いることができます。あるパーティション内でクロックのカウントダウンにより生成したタイミングで、他のパーティションの実行を制御することもできます。

FMU 内のモデルパーティションは、実行の優先順（プライオリティ）を定義することができます。また、プリエンプションにより実行中のモデルパーティションを一時中断し，よりプライオリティの高いモデルパーティションの実行に切り替え実行を行うことも可能です。

2.5.2. SE モードの FMU の実行例

図 2.5.2 は、3 つのモデルパーティションを持つ FMU を、横軸を時刻として、3 つのタスクの実行の様子を示したものです。ここでは 3 つのモデルパーティションを「タスク 1」「タスク 2」「タスク 3」と呼ぶことにします。「タスク 1」は 10[ミリ秒]の周期クロックで駆動します。「タスク 2」は、非周期クロックでの駆動として、タスク 1 の中でクロックの計数によりタスク 2 を開始するクロックのタイミングを生成します。「タスク 3」は 50[ミリ秒]の周期クロックを用いて駆動します。また，プライオリティは，タスク 1 が最も高く、次にタスク 2、タスク 3 が最も低いプライオリティを割り当てます。

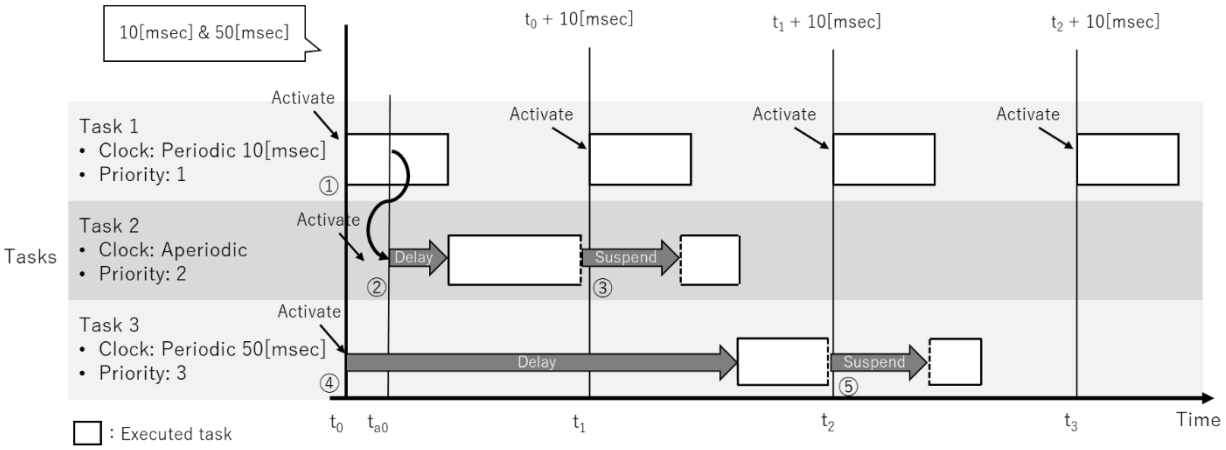


図 2.5.2　SE モードの FMU の内部動作の一例

（FMI 仕様書：Functional Mock-up Interface Specification_Version 3.0 より引用）[7]

① 時刻 t0 で、タスク 1 とタスク 3 のクロックに対して、10[ミリ秒]と 50[ミリ秒]が同時に経過し、タスク 3 に対してプライオリティが高いタスク 1 がアクティベートされます。

② タスク 1 の実行中にタスク 2 のクロックがトリガされますが，タスク 1 に対しタスク 2 はプライオリティが低いので、タスク 2 はタスク 1 の終了まで実行の開始を待ちます。タスク 1 が終了するとタスク 2 が開始します。

③ タスク 2 はタスク 1 のクロックのトリガ時刻（t1）までに終了しないので、タスク 2 は実行をサスペンドし、プライオリティが高いタスク 1 が開始します。タスク 1 の実行が終了した後、タスク 2 は実行を再開します。

④ タスク 3 は，t0 でトリガされますが、高プライオリティのタスク 1 とタスク 2 の実行が終わるまで開始を待ちます。タスク 2 が終了した後，タスク 3 が開始します。

⑤ 時刻 t2 でタスク 1 が開始すると、タスク 3 は実行を中断します。タスク 1 の終了後，タスク 3 は再開します。

2.6　FMI の互換性に関する一般的な注意点

2.6.1　FMI のバージョン違いと動作モード違いによる互換性

FMI には 1.0、2.0、3.0 の三つのバージョンが存在しています。それぞれの間での FMU の互換性はありません。このため FMU を作成するツール環境と、FMU を読込んで実行する側のツール環境が 1.0、2.0、3.0 のいずれのバージョンに対応しているのかを確認して、どのバージョンを使ってモデルを交換するかを決める必要があります。

また、ツール環境によって、Model Exchange (ME)、Co-Simulation (CS)、Scheduled Execution (SE) の FMI の 3 つの動作モードのどれに対応しているかには違いがあります。FMU を作成する側と FMU を読込んで実行する側で、どの動作モードの FMU を使用するかも確認しておく必要があります。

なお、FMU を作成するツール環境には、複数の FMI のバージョンおよび動作モードに対応しているものも多くあり、FMU を作成する際に FMI のバージョンと動作モードを選択できます。また、ME と CS の両方のモードに対応した FMU を作成できるツール環境も存在します。

一方、FMU を読込んで実行するツール環境には、バージョンの異なる FMU を混在させて実行できる環境、あるいは、ME と CS の両方のモードの FMU を混在して実行できる環境も多く存在します。

全体的には、FMI1.0 は旧式の規格になるため、FMI2.0 を利用することが推奨されます。FMI3.0 は規格が新しく対応するツール環境が少数に限られているため注意が必要です。

2.6.2　FMU の動作環境の OS 互換性

FMU 内部には「binaries」と呼ばれるフォルダがあり、その中で win32、win64、linux32、linux64 などの OS と bit 数を組み合わせたバイナリコードを保存するサブフォルダが存在します。取り込み実行する側のツールにより差はありますが、通常ではこれらのツールは決められた OS、bit 数で無ければ実行できません。このため予め OS と bit 数を決めて FMU を作成する必要があります。なお binaries 以下のサブフォルダは複数持つことが認められているので、FMU の中に複数の実行可能なライブラリファイル（Windows の dll ファイルあるいは Linux の so ファイル）を保存することが可能です。

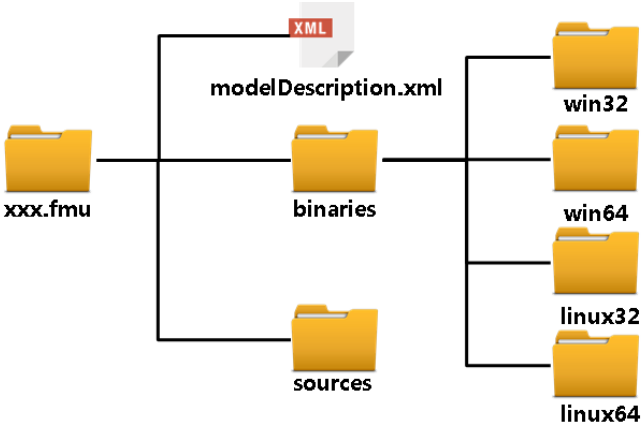


図 2.6.1　FMU の構造

2.7. FMI の情報隠蔽に関する一般的な注意点

2.7.1. 内部変数の観測

FMU の生成ツールにより隠蔽の程度が異なりますが、入力・出力・パラメータ・開示変数しか外部から見ることができません。このため観測が必要な変数は意識的に外部から参照できるようにする必要があります。

2.7.2. ソースコードの受け渡し

FMU にはソースコードを格納するフォルダ「sources」があり、この中にソースコードを格納することができます（図 2.7.1）。ソースコードの格納は任意ですので情報の隠蔽性を考える場合には、ソースコードを格納すべきではありません。一方で、OS 互換性のところで述べたように、実行可能な形式を binaries に保存して提供されないと実行できません。例えば ARM プロセッサ上で FMU を動作させたい場合、大半のツールで実行可能な FMU を作成できません。このような場合には、ソースコードを内部に持たせて作成し、取り込み側ツールでコンパイル、リンクを行います。

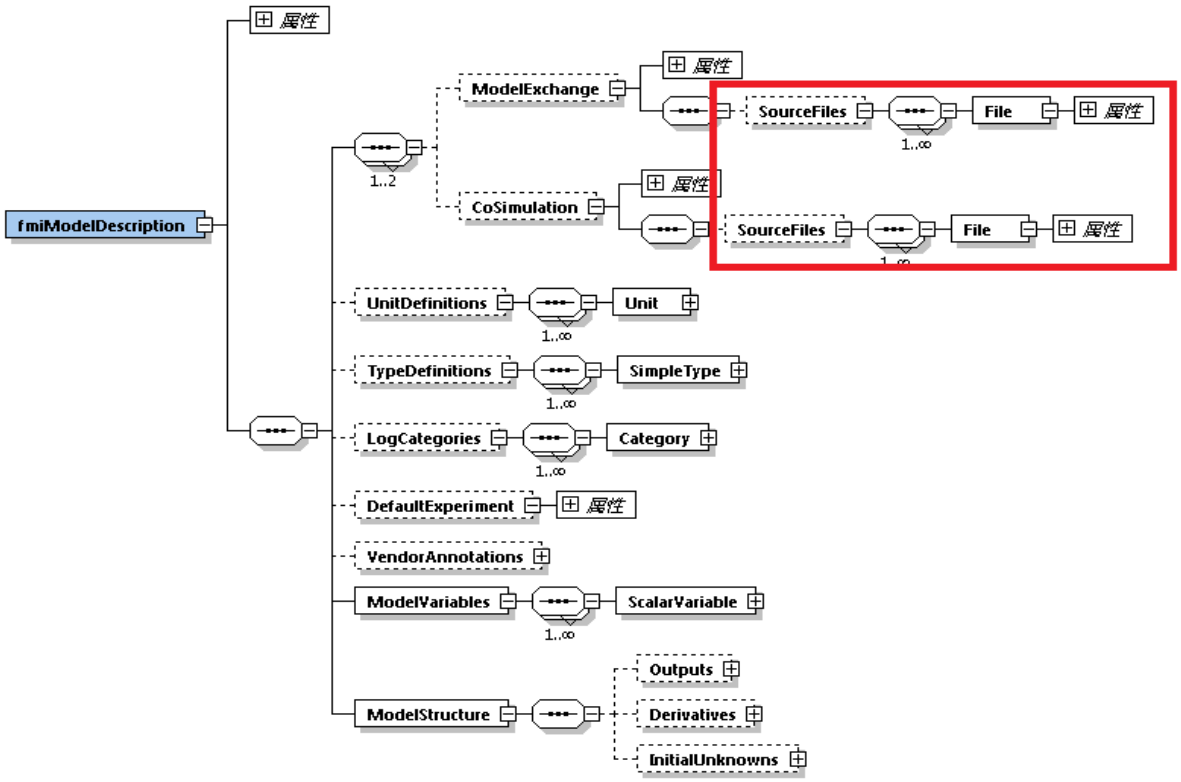


図 2.7.1　modelDescriptionFile の構造（対象部分のみ構成展開；赤色枠内）

（FMI 仕様書：FMI_for_ModelExchange_and_CoSimulation_v3.0 より引用）[7]

2.8. FMI を利用するツール環境に関する一般的な注意点

2.8.1. FMU の実行ライセンス

FMU を動作させるには、その FMU を生成したツールによっては、その生成環境が指定した実行用ライセンスを必要とするものがあります。ライセンスが必要な場合には、単に FMU を渡すだけでは使用できないので、生成側ツールの設定、取り込み側ツールのライセンス環境などを確認して下さい。

2.8.2. FMU の実行環境

FMU 内には、2.6.2 で述べたように取り込み側ツールの OS、bit 数に合ったものを生成する必要があります。また、CS の FMU としては、内部にソルバを持つものが多く利用されていますが、FMU 自身は通信の機能だけを持ち、シミュレーションにはソルバ自身を利用できる環境が必要となる場合もあります。この場合は、ライセンス環境だけではなく、ソルバ自身がインストールされた環境が必要です。

2.8.3. パラメータ変更

FMI の規定では FMU の中に実行前に変更可能な値（数値、論理値、文字列）をパラメータとして定義することができます。取り込み側ツールではその値を変更できないものもあります。取り込み側ツールで変更できない場合 FMU を再生成して渡さざるを得ないので、実際に使用する値を生成時に取り込み側と決めておく必要があります。なお FMU の中の xml ファイル中にデフォルト値が書かれていますが、この値を変更してもパラメータの値は反映できません。

2.8.4. 名称

FMI では名称ルールが決まっています。FMU そのもの名称や、変数の名称（入出力、パラメータ）などで使用可能な文字列は Unicode の文字列とされています。

ツールに取り込む時に不要なエラーを発生させないためには、アルファベットで始まり、アルファベット、数字と“_”での組み合わせを使用することをお勧めします。

2.9. FMI 公式ホームページについて

FMI 公式ホームページ（https://fmi-standard.org/）について説明します。

2.9.1. FMI の規格文書

FMI 公式ホームページから、これまでに発行された FMI 1.0 から 3.0.1 までの「Complete Package」と「Specification」の二種類の規格文書を取得できます。

「Specification」は規格の仕様書だけですが、「Complete Package」には規格の仕様書に加えて、XML 形式ファイルのスキーマ定義ファイルと、共通ヘッダファイルが含まれています。スキーマ定義ファイルは、XML ドキュメント内のデータが FMI 規格に準拠したフォーマットや値になっていることを確認する際に使用できます。共通ヘッダファイルは FMU ファイルを実行する際に利用する API 関数が定義されています。この二種類のファイルは、FMU ファイルを自身で作成する・加工する、あるいは、FMU ファイルの内部情報を読み出すようなツールを自作する際に有用です。

2.9.2. FMI 規格対応ツールを調べる

FMI 公式ホームページの「Tools」で、FMI 規格に対応したツールを検索することができます。

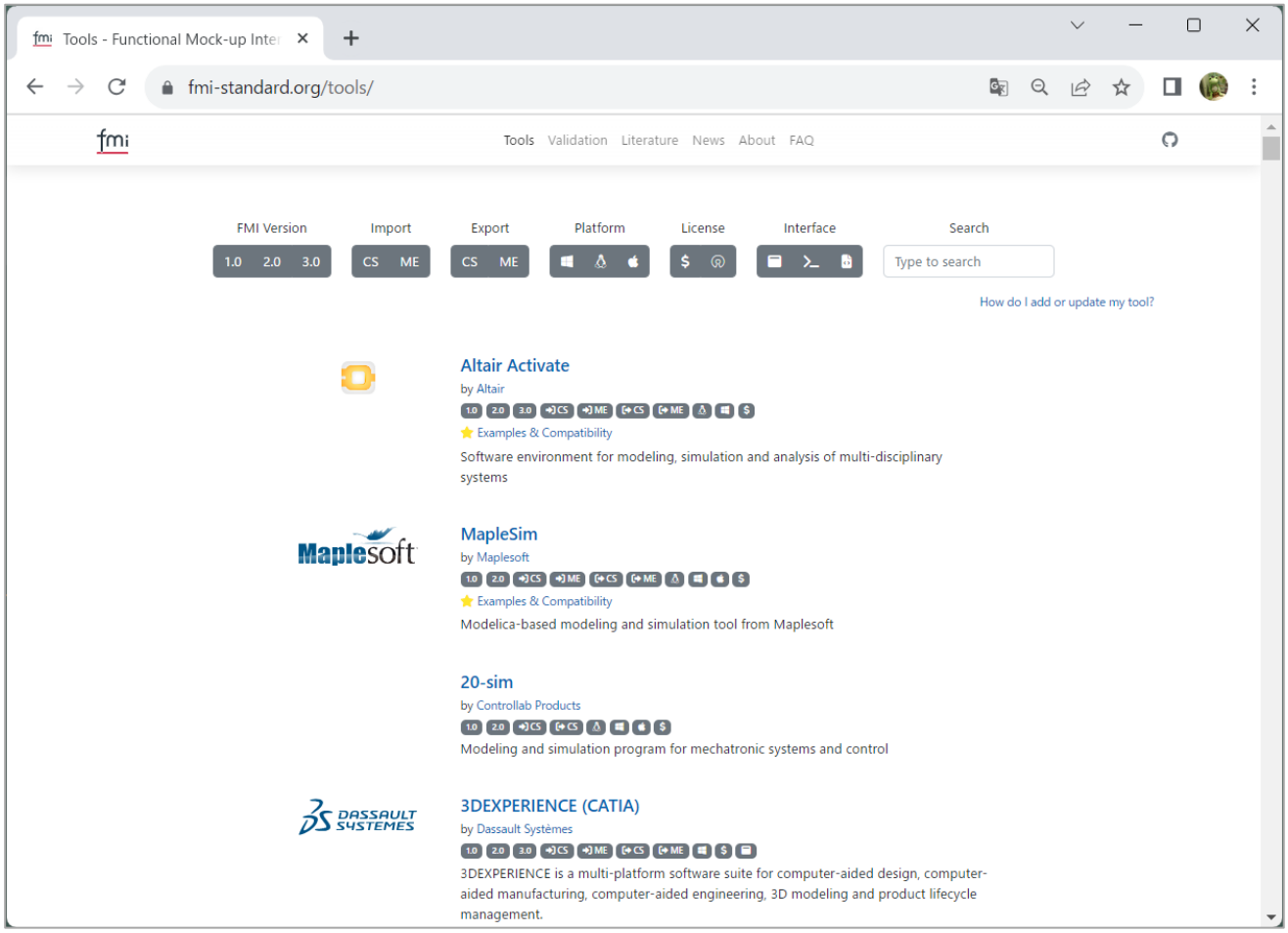


図 2.9.1　FMI 規格に対応したツールの検索機能（FMI 公式ホームページ）[10]

この画面ではフリーキーワードを使って FMI 規格対応ツールを検索することができるとともに、アイコンを使って各ツールの FMI 規格対応状況が分類されているので、利用者の環境や用途に応じて、利用可能なツールを簡単に調べることができます。

- FMI 1.0/2.0/3.0 のどの FMI 規格に対応しているか
- CS（Co-simulation モード）あるいは ME（Model Exchange モード）の FMU をインポートできるか
- CS（Co-simulation モード）あるいは ME（Model Exchange モード）の FMU を生成（エクスポート）できるか
- 稼働環境（Windows、Linux, iOS）
- 有償ツール／無償ツールのいずれか
- ツールのユーザインタフェースが、GUI 方式、コマンドライン方式、API ライブラリのいずれか

公式ホームページは日々更新されていますのでご確認下さい。

2.10　FMI に関する技術活動およびイベント

2.10.1. Modelica Conference

FMI は、Modelica Association Project (MAP)[19]の一つとして、仕様改訂および展開活動が成されているため、Modelica Conference[20]の主要議題の一つとなっています。Modelica Conference は、奇数年に隔年毎に欧州で開催され、FMI[21]、およびその派生プロジェクトである SSP (System Structure and Parameterization)[22]、DCP (Distributed Co-Simulation Protocol)[23]、eFMI (Functional Mock-up Interface for embedded Systems)[24]などの他の MAP の最新動向などと共に、幅広い Modelica 言語関連のテーマについて論文発表と議論が行われています。また、偶数年には、北米およびアジアで、American Modelica Conference と Asian Modelica Conference がそれぞれ開催され、補完的な役割を果たしています。各 Modelica 関連情報の詳細については、Modelica Association のホームページ[1]から閲覧することができます。

図 2.10.1 に、Modelica Conference および Modelica Association Project の推移について、示します。

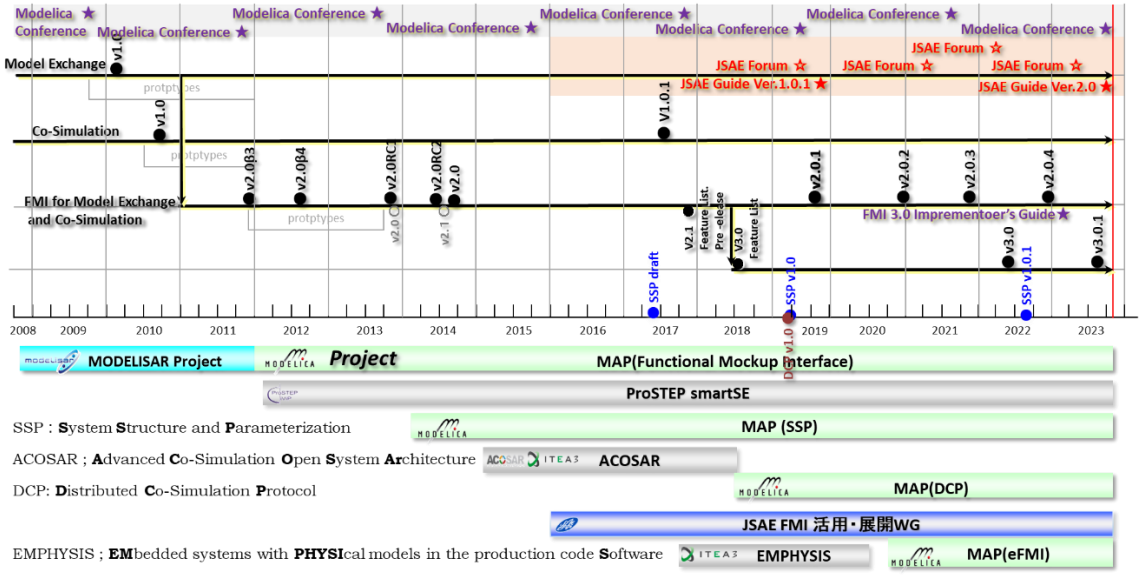


図 2.10.1 Modelica Conference および Modelica Association Project の推移

2.10.2. prostep ivip SmartSE Project

prostep ivip[25]は、ドイツを中心とした欧州の産業界で、デジタル開発を活用した開発業務の効率化に関する会議体の主催や技術標準の発行、手法の啓発・展開を企図している業界団体です。その対象としては、3D-CAD のデータ標準化、MBSE(Model Based System Engineering)のための要求仕様記述手法や管理手法の標準化、MBD(Model Based Development)のためのモデル交換・接続仕様やモデル流通管理手法の標準化などがあります。その中で、Smart Systems Engineering (SmartSE) Project[26]は、主に自動車業界での MBD 推進のための活動にフォーカスしたプロジェクトです。SmartSE Project では、モデル間の接続仕様として FMI を推奨しており、モデル接続・交換のためのガイドラインを発効しています。最新バージョンは、Smart Systems Engineering Collaborative Simulation-Based Engineering Version 3.0[27]であり、prostep ivip の Web ページからダウンロード可能です。

2.10.3. FMI Implementers' Guide

FMI Implementers’ Guide は、FMI によるモデルインポート、エクスポートをツールに実装する際

のガイドブックであり、現在は、FMI Ver. 3.0 に対応した FMI 3.0 Implementers' Guide[28]が、Modelica Association Project FMI と prostep ivip の共同作業により公開されています。

2.10.4. その他

FMI の全バージョンの詳細仕様書は、FMI Project のホームページ[20]からダウンロード可能です。また、最新の FMI 対応ツールの一覧もホームページ[29]から参照できます。

FMI と Python 環境との連携も進んでおり、フリーライブラリとしては、FMPy[30]や PyFMI[31]があります。

第 3 章　FMU の動作モード選択のガイドと各工程の流れ

この章では、Model Exchange と Co-Simulation についてもう少し詳しく説明します。
FMI には Model Exchange、Co-Simulation、Scheduled Execution の三つの方式がありますが、Scheduled Execution は FMI3.0 より追加されたもので、対応ツールが限定されているため、ここでは割愛いたします。3.1 では、それぞれの特徴を知ることで適した手法を選択する指針を解説しています。3.2 では、モデル作成からシミュレーション実行までの工程について、3.3 及び 3.4 では、トラブルが発生した際の問題解決への糸口を見つけるため、それぞれの工程で起きうる問題と対策について解説しています。

3.1.　FMU の動作モード選択の指針

Model Exchange、Co-Simulation それぞれの特徴を表 3.1-1 にまとめました。表の最後に、当ワーキンググループ推奨の用途をまとめました。Model Exchange が有用な場合としては、制御モデルの交換が挙げられます。その理由としては、制御モデルは一般的に代数ループを含まない点、通信遅延が無い事が望まれる点、モデルが一般的に陽解法固定タイムステップソルバで解け、ソルバとモデルの相性問題が起こりにくい点が挙げられます。Co-Simulation を用いる場合は、それぞれ適したツールでモデルとソルバの相性が確認された FMU 同士をつなげることが可能であるため、複合物理システムモデルに有用と言えます。また、各 FMU が個別のプロセスを持つことが可能なため、並列計算を行いシミュレーションの計算速度向上を図る事も期待されますし、マルチレートでの実行も可能です。Hardware in the Loop Simulation を行う際にも、モデル作成ツールや、実行ハードウェアに依存しない流通が可能です。

表 3.1-1　Model Exchange／Co-Simulation の特徴

	Model Exchange	Co-Simulation
代数ループを含んだ FMU	FMU 生成不可能、もしくは安定性が損なわれるため、非推奨	生成元ツールのソルバにより安定した計算が可能 計算時間増大の可能性
通信遅延	発生無し	Communication Step Size の適切な設定が必要
可変タイムステップソルバ使用時	モデル／ソルバ相性の確認が必要	モデル／ソルバ相性は FMU 生成時に確認済
陽解法固定タイムステップソルバ使用時	モデル最大固有値の抑制が必要	モデル最大固有値の抑制が必要
Core 分割	不可能 モデル全体で単一プロセス	可能 FMU 毎にプロセスを生成可能
推奨用途	制御モデル	複合物理システム 大規模モデルの並列計算処理 マルチレート Hardware in the Loop Simulation

3.2.　FMU 作成から、交換、シミュレーション実行までの流れ

シミュレーションを成功させる上で、トラブル時の原因究明を行うには、どのような流れで FMU 生成から、交換、シミュレーション実行が行われているかの理解が役に立ちます。大まかな流れを、図 3.2.1 にまとめます。

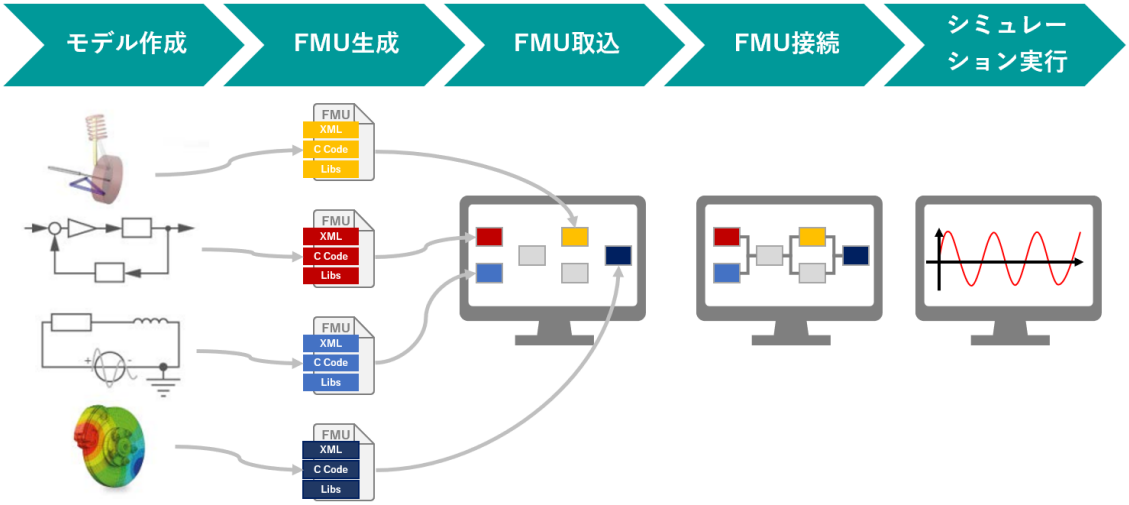


図 3.2.1　FMU 生成から、交換、シミュレーション実行までの流れ

(図の一部は 13th International Modelica Conference 資料より引用) [32]

3.3. それぞれの工程で起こりうる問題と対策 (Model Exchange)

3.3.1. モデル作成時

Model Exchange にて、FMU を作成するには出力するモデル内部に代数ループが無い事が推奨されます。作成したモデルに代数ループがある場合は、その原因特定と対策を行い陽的なモデルに変更しましょう。

例を見てみましょう。図 3.3.1a のモデルでは、可変のオリフィスを PI 制御で目標の流量に達成するように開口径を変化させていますが、図 3.3.1b のハッチング部に代数ループが発生しています。

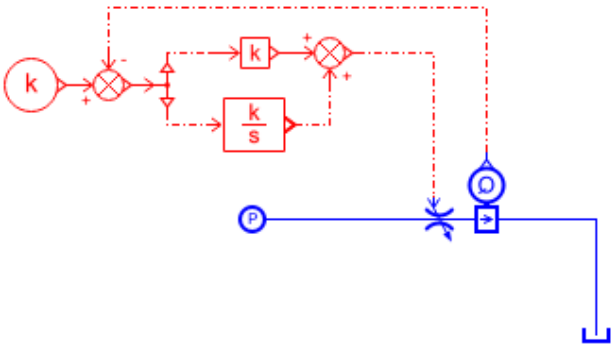


図 3.3.1a　流量の PI 制御例

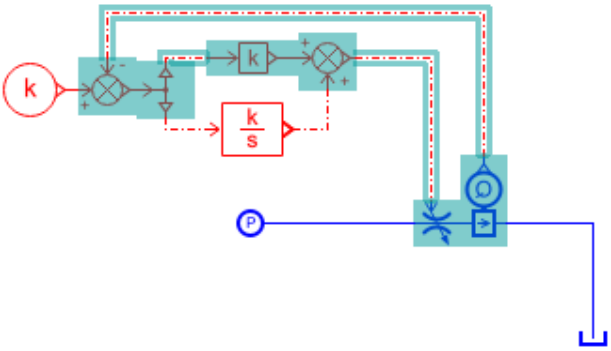


図 3.3.1b　流量の PI 制御例 代数ループ発生個所

図 3.3.2 のように、一巡ループ間に積分計算を挿入することで代数ループを解消することが出来ます。Actuator の時間応答遅れを模擬し、出力に 1 次遅れを挿入することが一つの対策として考えられます。

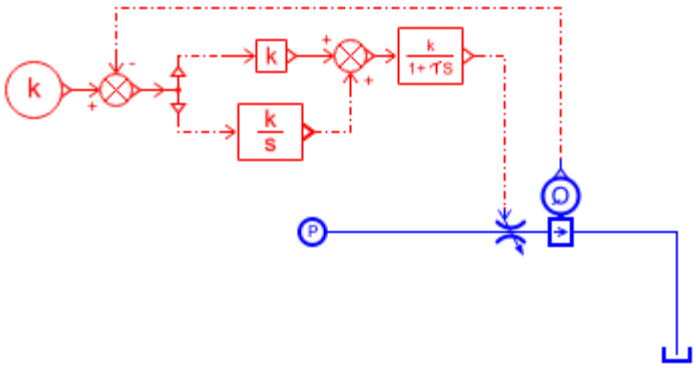


図 3.3.2　1 次遅れを挿入

このように物理モデル内に PI 制御を用いている場合に発生することがある点に注意しましょう。また、物理モデルのみで代数ループが発生する場合もあります。電位一定、変位一定などの拘束条件を持つ計算素子を用いた場合は FMU 生成前に代数ループの有無を確認しましょう。

3.3.2. FMU 生成／取込時

FMU のパラメータや状態変数名、入出力の情報などは、modelDescription.xml というファイル名の model Description File に格納されています。これらの情報を正しく書き出し、読めるかどうかが FMU 交換成功の鍵です。FMI Specification にどのように model Description File を記述すべきかが書かれており、また Compliance Checker が FMI ホームページ[13]にて公開されており、FMU 出力に対応しているツールベンダはこのチェッカにて標準対応しているかを検証しています。

3.3.3.　FMU 接続時

FMU の入出力を正しく接続できるかが、最初の関門です。別紙のチュートリアルの例題では、2 つの FMU 間の接続は 1 つの組み合わせしかなく、問題なく接続ができました。しかし、複数の入出力がある FMU を取り扱う場合は、どの入出力を接続するか確認する必要があります。様々な制御や物理モデルをやり取りする場合に、FMU 間でやり取りすべき変数は何か、また単位はどうすべきかを判断するには、参考例として、MBD 推進センター(JAMBE の)「自動車開発におけるプラントモデル I/F ガイドライン（ver4.0)」［34］が活用できます。これらを元に入出力ルールを、あらかじめモデル交換者間で合意する必要あります。この点については、4 章で詳しく述べます。

Model Exchange 接続時に発生しうる問題としては、FMU を接続し、入出力の繋がりで循環がある場合、本当は存在しない不必要な代数ループが発生することがあります。その場合、model Description File の Dependency をユーザにて変更し、代数ループの解消を行う事も考えられます。この点については 4.3.3 で事例及び対策例を紹介します。

3.3.4.　シミュレーション実行時

Model Exchange では、取込側のツールのソルバを用いて計算を行います。シミュレーションがうまく流れるかは、モデルとソルバの相性が良いかと言い換えることができます。この相性を理解するには、シミュレーションの安定性に関する知見が役に立ちます。ここでは、簡単に技術的内容をご紹介します。

FMI を用いて交換されるモデルは、主に時間に関する常微分方程式のみで表されています。

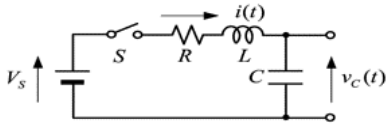


図 3.3.3　LRC 回路

$$L\frac{d^2q(t)}{dt^2} + R\frac{dq(t)}{dt} + \frac{1}{C}q(t) = V_S \qquad (1)$$

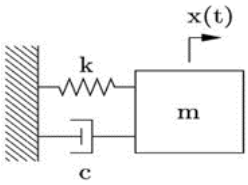


図 3.3.4　単振動システム

$$M\frac{d^2x(t)}{dt^2} + C\frac{dx(t)}{dt} + Kx(t) = F_{ext} \qquad (2)$$

このように様々な物理領域において、モデル化対象の振る舞いが式(1)、式(2)のように 2 階の常微分方程式で表されます。この形は図 3.3.3 の LRC 回路や図 3.3.4 の単振動システムのように固有値や共振周波数を持ちます。解いているモデルの固有値が使用しているソルバの収束領域に入っているということが、発散しない、解けるシミュレーションが行われている状態です。

また、詳細な物理モデルを取り扱うと、線形時不変ではなく非線形なシステムであることがほとんどです。その際は、適切な時間刻みやソルバ次数などを変更しながら解いていく、可変タイムステップソルバが用いられることがあります。可変タイムステップソルバは、シミュレーションツールそれぞれで工夫がなされており、ツールで作成されるモデルを解きやすいようにチューニングされています。その為、あるツールで作成した FMU を Model Exchange で出力し、他ツールで取込み、シミュレーション実行する際にうまく収束しない、もしくは結果に差異が生じるなどの問題が発生する可能性があります。これが、モデルとソルバの相性の問題です。

3.4. それぞれの工程で起こりうる問題と対策 (Co-Simulation)

3.4.1. モデル作成時

Co-Simulation では、モデルとソルバを合わせて出力します。そのため、仮にモデル内で代数ループがあったとしても問題ありません。拘束方程式を使用した場合の機構や電気回路モデル、もしくは 3D-CFD などの特殊なソルバが必要なモデルを使用する際に活用できます。

3.4.2. FMU 生成／取込時

FMU のパラメータや状態変数名、入出力の情報がうまくやり取りできるかは、Model Exchange と同様の問題を持ちます。

3.4.3. FMU 接続時

FMU 入出力の整合は、Model Exchange と同様ですが、Co-Simulation 時に不必要な代数ループが発生することはあまりありません。なぜならば、Communication Step Size による通信遅れがあるために代数ループが解消されるからです。ほとんどの取り込み側ツールでは、Co-Simulation 使用時には FMU を接続し、入出力の繋がりで循環がある場合でも、代数ループとみなさず処理が行われます。

3.4.4. シミュレーション実行時

Co-Simulation を行う際は、3 通りの方法があります。1 つ目は、ソルバとモデルを合わせて出力する Coupling with System Models もしくは単純に Stand Alone と呼ばれる手法です。2 つ目及び 3 つ目は Tool Coupling、Distributed と呼ばれているもので、ツール間のインターフェースの役割のみを果たし、双方のソフトウェアを立ち上げて行う Co-Simulation です。手法の違いによって、実行時に必要なソフトやインストール環境が異なる点に注意してください。

計算結果において、発散する、もしくは計算速度が遅くなる場合があります。Co-Simulation での Communication Step Size により生じる問題です。また、シミュレーション結果はこの Communication Step Size 設定によって差異が生じます。

Co-Simulation では、マスタ、スレーブ間で決められた時間間隔で通信を行います。通信間隔の間は、各 FMU の入力値が一定と扱われるため、ステップ応答遅れが FMU 間に生じます。一部のツールでは、連続的な入力となるように補間を行う場合もありますが、このステップ応答遅れが挿入されることで下記のような課題が起こりえます。

1) 通信遅れにより、接続したモデル系全体での安定性が損なわれ、発散リスクが高まります。通信間隔が大きい程問題となるため、この視点からは、細かい間隔が望ましいです。
2) また、通信間隔を小さくすればするほど、ステップ応答遅れが及ぼすモデル全体系への影響も小さくなるため、計算誤差の低下も期待できます。
3) しかし、細かい通信間隔を設定すると通信時の不連続点が増大し、ソルバでの収束演算回数が増すことで計算時間が増大してしまいます。

通信間隔をなるべく大きくとれるよう、モデル間の連成が弱い点での接続を行うことが望ましいです。また収束計算演算数を抑えるために、陽解法固定タイムステップソルバを使うなどの対策が有効ですが、そのモデルの固有値がソルバ収束領域内にすべて入っており、発散しないことを確認する必要があります。

なお、利用するツールによって機能の有無がありますが、コンピュータに複数コアまたはスレッドがある場合、Co-Simulation では FMU 毎に別個の executable を実行できるので分散処理が可能です。これに対し Model Exchange では、基本的にモデル全体で 1 つの executable を実行するので、分散処理には向いていません。この機能をうまく活用すれば計算速度の向上が期待出来ます。

第 4 章　実務適用のために知っておきたいこと

この章では、FMI を実際に使うために FMU 生成上の注意点から読込実行までのエラーなどについて説明を行います。4.1 では、プラントモデルの入出力 I/F を決定する上で指針としている JAMBE ガイドラインについて解説します。4.2 では、様々な部類のモデルを分割する際の注意点を挙げています。4.3 では、FMI を利用する上で起こりうるエラーについて解説しています。

尚、一般の異種ツール間でのモデル連携利用における注意点と FMI 特有の注意点を区別するため、【共通】、【FMI】を表記しています。

4.1.　プラントモデルの入出力決定の指針

複数のモデルをひとつの全体モデルにまとめる場合、モデルの入出力関係が重要になります。FMI を使用した場合も同じです。4.2 ではプラントとコントローラを 4 つの関係に分けて接続・分割についての注意点を説明しますが、その前に JAMBE （MBD 推進センター）のホームページで公開されている、「自動車開発におけるプラントモデル I/F ガイドライン（以下、JAMBE ガイドライン）」を確認しておきます。ここではまず提示された 5 つの原則を簡単に説明し、この原則と FMU 化の対象となるモデルの入出力について説明します。

表 4.1-1　JAMBE ガイドラインの原則

基本原則	
第一	プラントモデル間はアクロス変数とスルー変数でつなぐ。 また、アクロス変数とスルー変数の向きは互いに逆向きとする。
第二	エネルギソースからエネルギシンクへ流れる方向をエネルギの正の向きとする。
第三	スルー量・アクロス量を蓄積する要素を基準として、全体の I/F を考える。
第四	スルー変数の正負は、エネルギの正の流れの向きと、スルー変数の入出力の向きが同じとき、正とする。
第五	入出力の単位は SI 単位系、SI 組立単位系を利用する。 量記号は JIS 規格を使用する。

(自動車開発におけるプラントモデル I/F ガイドライン(ver.4.0)より引用)

4.1.1.　接続信号の選択【共通】

JAMBE ガイドラインでは第一原則、第三原則の適用部分です。第一原則によれば、プラント間の接続では、図 4.1.1 の様に一方のプラントがスルー量入力の場合、そのプラントはアクロス量を出力します。接続先のもう一方のプラントはそのアクロス量を入力として受け取り、はじめのプラントが受け取るスルー量を出力することになります。第三原則では表 4.1-2(a)、表 4.1-2(b)のように接続部位において表現されている物理的な部品を 2 つに分けて考えます。

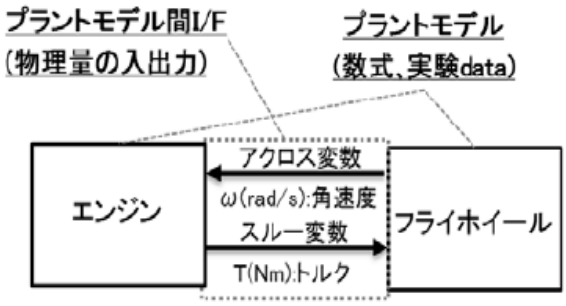


図 4.1.1 第一原則（JAMBE ガイドラインより引用）

表 4.1-2 (a)　スルー量を蓄積する要素（スルー変数を入力とする）

物理領域	機械（並進）	機械（回転）	電気	熱
物理的な部品	質量	慣性モーメント	電気容量（キャパシタ）	熱容量
入力：スルー変数	力	トルク	電流	熱流量
出力:アクロス変数	速度	角速度	電圧	温度

表 4.1-2(b)　アクロス量を蓄積する要素（アクロス変数を入力とする）

物理領域	機械（並進）	機械（回転）	電気	熱
物理的な部品	バネ	ねじりバネ	電気誘導（インダクタ）	該当なし
入力：アクロス変数	速度	角速度	電圧	該当なし
出力：スルー変数	力	トルク	電流	該当なし

例として機械（回転）で考えてみると次のようになります。
慣性モーメントはスルー量（トルク）を蓄積します（表 4.1-2a)。このため慣性モーメントがモデルの接続部についている場合は、トルクが入力で角速度（アクロス量）を出力とすることが推奨されています。非因果ツールで速度入力の FMU を生成しようとした場合には生成時にエラーが発生し、回避するためには加速度入力に変更する必要があります。しかし加速度を生成するためには微分が必要になり数値シミュレーション的に不安定になりやすいと考えられています。このため JAMBE ガイドラインではアクロス変数ではなくスルー変数を入力することを原則としています。なおプラントモデルが非因果モデリングツールを用いて作成されている場合、自動車技術会「非因果モデリングツールを用いた FMI モデル接続ガイドライン Ver.1.0」[4] で説明されている非因果アダプタを適切に用いればこの原則とは異なる接続を実現することができる場合もあります。

JAMBE ガイドラインに基づく接続では、慣性モーメント同士の接続ができません。この場合、慣性モーメントをどちらかにまとめるのが好ましいと当 WG では考えます。やむを得ず両側に慣性モーメントを持つ形で分ける場合には、いずれかに回転バネ要素を入れることでスルー変数入力からアクロス変数入力に切り替える必要が生じてしまいます。この場合、回転バネと慣性モーメントの大きさを適切な関係にしないと、小さな時定数を持つシミュレーション時間が長い不安定なモデルになってしまいます。

ねじりバネが接続部にある要素ではこれとは逆にアクロス量（角速度）を入力することができます。スルー変数であるトルクを出力とすることができます。

なお JAMBE ガイドラインでは、減衰（ねじりダンパ）が接続部にある場合は、スルー変数入力でもアクロス変数入力でも構わない、としています。機械系では実際に減衰のみを接続部に持つことはあまり見

受けられません。電気系では電気抵抗がスルー変数入力・アクロス変数入力の両方に対応し、これは十分に考えられる接続です。

4.1.2. 信号の取り決め【共通】

4.1.2.1. 入出力の符号

第二原則、第四原則が適用される部分です。JAMBE ガイドラインでは第二原則の例として、エンジン側から駆動系、走行抵抗に向かうエネルギの流れ（仕事率、またはパワー）を正と定義しています。これに従うと変速機からデフへトルクを入力する（デフから変速機へ角速度を入力する）という関係は、加速時に正のトルクをデフへ送る、減速時には負のトルクをデフへ送るという関係になります。

4.1.2.2. 単位系

第五原則で単位系は SI 単位系としています。FMI では modelDescription.xml に単位系を記載し内部で明示的に持つことができます。しかし FMI の規格において単位系を利用する機能は必須項目ではないため、生成された FMU 内に必ずしもこの記述があるとは限りません。また取り込み側でその単位系定義を使用しているとも限りません。

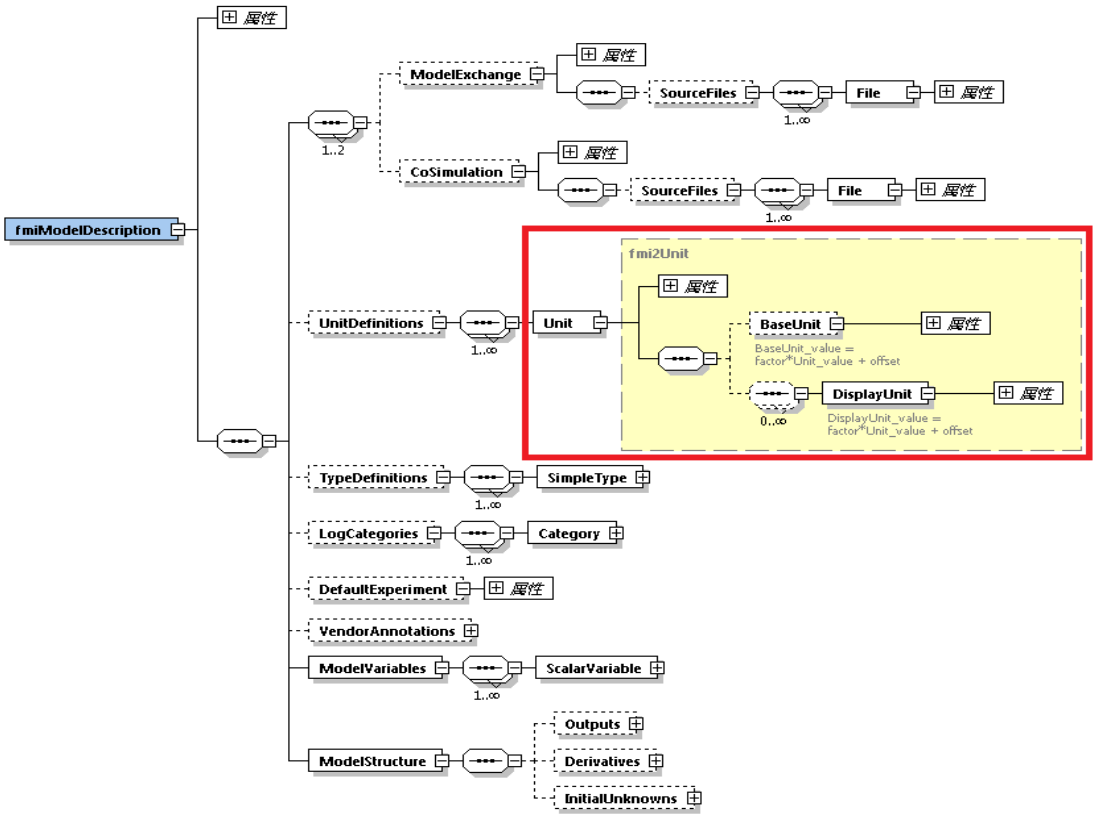


図 4.1.2　modelDescriptionFile の構造（対象部分のみ構成展開；赤色枠内）

（FMI 仕様書：FMI_for_ModelExchange_and_CoSimulation_v3.0 より引用）[7]

自動車開発においては、回転数（回転速度）を表すのに[rad/s]ではなく [rpm]を単位として用いることが多く見受けられます。またしばしば温度は摂氏[℃]で表現されます。このように単位系は誤解を招きやすいため、JAMBE ガイドラインではモデルそのものとは別情報の「サブシステム I/F 定義書」として授受することが提案されています。当 WG でもモデルである FMU の外で伝達することが望ましいと考えます。

4.2. モデルの分割における注意点

前節でプラントモデルの分割について触れました。ここではサブモデル作成上について 2 つの接続先をコントローラとプラントに分けて分割の説明をします。

4.2.1. コントローラとコントローラ【FMI】

アナログ（連続系）信号入出力とデジタル（離散系）信号入出力を分けて考える必要があります。ME を用いた場合、実際のアナログ信号はモデル間で遅れを伴わずシミュレーションが行われます。CS を用いた場合は、信号授受で通信間隔分の遅れが生じることを配慮しなければなりません。デジタル信号ではもともとサンプリング分の遅れがあるので、その遅れと CS の通信間隔との関係から影響の有無を考察しなければなりません。

CS の場合の手順：

1 つのコントローラで行われる処理は FMU を分けずに 1 つの FMU の中で行う。

通過するだけの信号は FMU を経由して送信しない。できる限り信号源となるコントローラと受信側コントローラを直接結ぶ。

FMI2.0 まではバス（またはベクトル）端子は存在していません（FMI3.0 ではバス端子をサポートしています）。コントローラ間を接続する場合、FMI2.0 では全ての信号に 1 つづつの接続を行うと多数の接続が発生してしまいます。必要な入出力に限定して接続を行うようにして下さい。特に入力端子については入力信号がないとエラーになるツールも多いので不要な端子を設けるべきではありません。

FMI の規約では出力変数(y) 以外に開示変数(v) が定義されています。読み込んだツール上で観測したい変数がある場合には FMU を生成する時点で(y) または(v) として定義する必要があります。一方、読込側ツールによっては(v) を観測できないものもあります。この場合、接続をしない変数も出力変数として設定を必要とする場合があります。

4.2.2. コントローラとプラント【共通】

4.2.2.1. コントローラとプラント間の信号

一般にコントローラとプラント間での送受信される指示値や観測値信号はケーブルを通じた電気信号ですが、回路としてモデル化されるケースは多くありません。厳密にいうと電気回路における信号の遅延を考える必要があります。どの部分でアクチュエータへの指示信号やプラントが出力するセンサ信号が離散化されるのかを考えてコントローラとプラントを分割する必要があります。特に CS の場合、前項でも取り上げたように遅延が正しく表現されているかを考える必要があります。

4.2.2.2. コントローラとプラント間の物理量

モデルの上では 4.2.2.1 で書いた信号と似ていますが、スルー変数、アクロス変数として実際に接続されている物理的な値のやりとりをする場合は 4.2.3 で示すプラントとプラントの接続と同じ状態になります。このため 4.2.2.1 で書いたような信号と同じなのかを判断する必要があります。

例えば図 4.2.1(a)のようにコントローラ (FMU1) からプラント (FMU2) のモータへの電流値を伝え、プラントから電圧が戻るような場合は、作成者側は FMU1 をコントローラとして考えたとしても、コントローラ自身にプラントの特性を持つため、結果は図 4.2.1(b)のような挙動を示します。

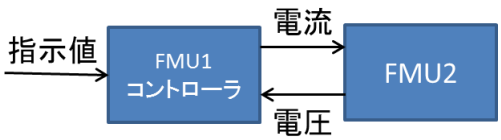


図 4.2.1(a)　コントローラとプラントのモデル例

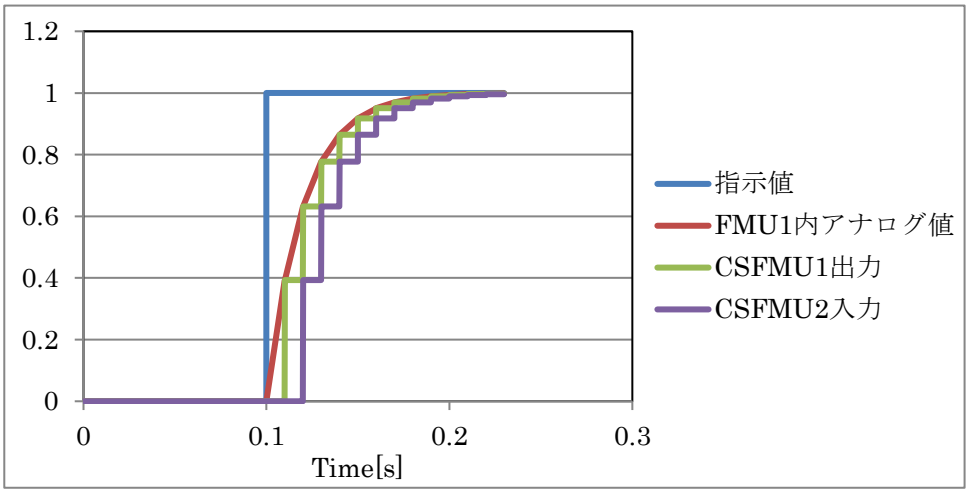


図 4.2.1(b)　CS における遅延の例　(通信間隔=0.01s)

4.2.3.　プラントとプラント【共通】

もっとも注意を払うべきなのはプラントとプラントの接続で機械的、電気的など物理的な接合を異なるモデルに分割する場合です。入出力については JAMBE ガイドラインに沿って考えることを当 WG では推奨します。FMU を読み込んだツールが非因果接続を認めるツールである場合には、4.1.2 で触れた非因果アダプタを用いる方法もあります。4.1 で説明したように JAMBE ガイドラインに従って分割・接続をする場合、スルー変数を入力とする部品同士、アクロス変数を入力する部品同士を直接接続することができません。

4.2.3.1.　モデル分割を避けたい位置

動作が速い（時定数が小さい）部位や、元々の動作の不連続性が高い部位での分割も可能な限り避けるべきと考えます。

・時定数が小さい部位の例：機械系では質量／慣性モーメントに対して並進／回転バネ定数が大きい。
・不連続性の高い部位の例：機械系では摩擦や突き当たり（衝突、機構的な可動範囲の制約）、電気系ではダイオードなどのスイッチング素子が該当します。

4.3. FMI とエラー

FMI を用いたモデルにおけるエラーも、FMI 特有のものと一般的なツール連携のものがあります。4.3.2 以降に各種エラーについて説明しますが、生成ツールで取り込むことができる場合、まず生成したツール自身で取り込みから実行まで実施し、問題ないことを確認するのが最初の確認事項だと考えます。

エラーメッセージは、取り込んだツールが出している場合と、FMU 自身が出している場合の二つの場合があります。どちらが出しているのかを確認できれば解決の糸口になります。FMU が出しているエラーは、その FMU を生成した際の元になるソースコードに記載されている可能性があります。

4.3.1. FMU の FMI 規格適合検証ツール【FMI】

FMU が FMI 規格に合わせて生成されているかを確認するツールとして Compliance Checker が存在します。Compliance Checker は FMI 公式ホームページにて実行ができます。図 4.3.1 は実行ページです。図 4.3.1 の「FMU Check」項目の「Validate your FMU ➔」をクリックすると、図 4.3.2 が表示されます。「Select」ボタンを押してチェックしたい FMU を指定します。サンプルモデルは自動車技術会ホームページの自動車制御とモデル部門委員会ページよりダウンロードできます。

ファイル名：Sample_3p1.zip　：ZIP ファイルから FMU_J1_Case1_Dymola_ME_2.fmu を取り出してチェックします。

結果の一例を図 4.3.3 に示します。問題の無いことが報告されています。ここで、「Model Info」タブをクリックすると、図 4.3.4 のように FMU のファイル情報が表示されます。また、「Variabes」タブをクリックすると、図 4.3.5 のように FMU に含まれている変数の情報が表示されます。「Files」タブをクリックすると、図 4.3.6 のように FMU に格納されているファイルの情報が表示されます。

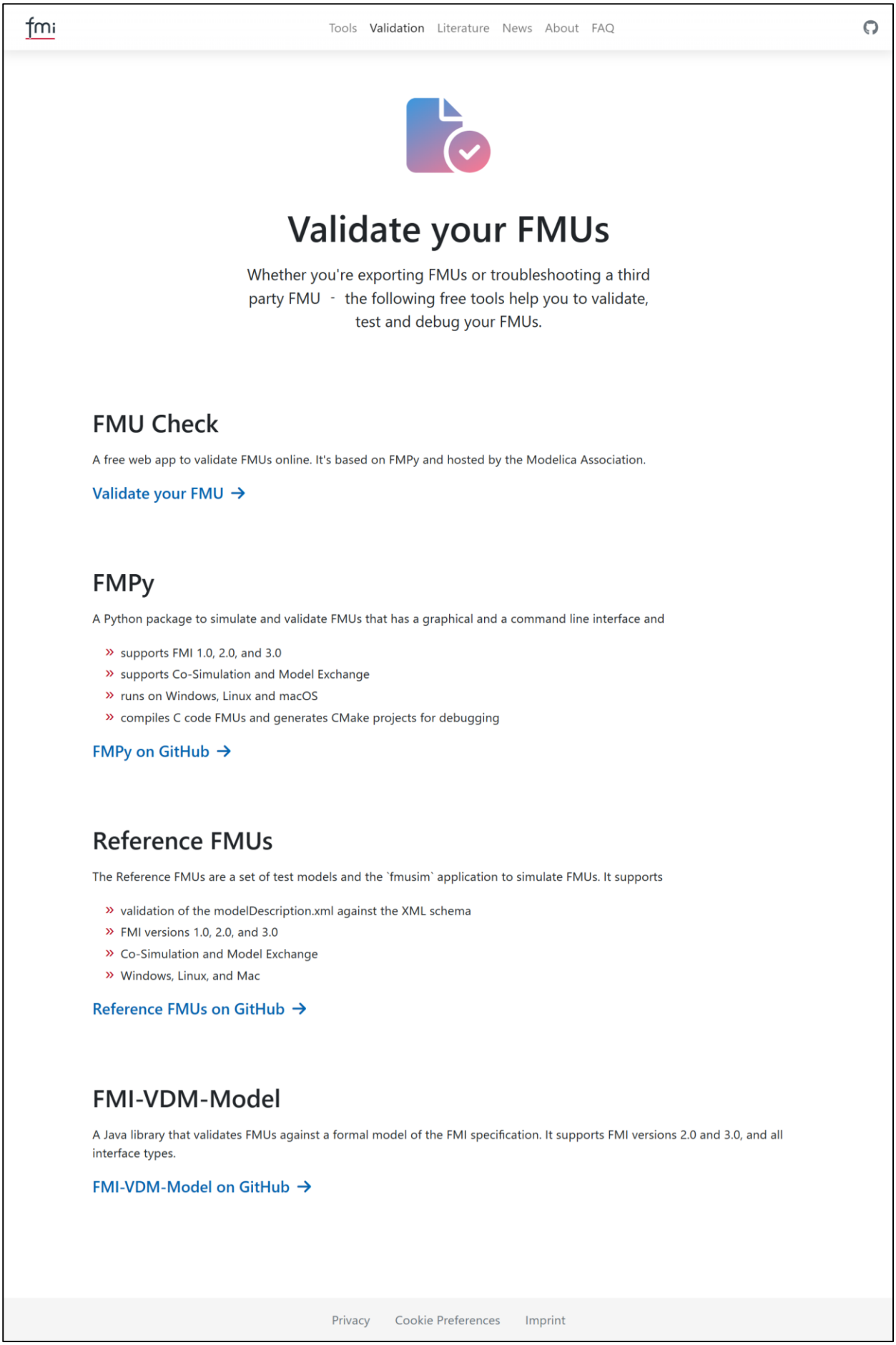


図 4.3.1　Compliance Checker の実行 Web ページ

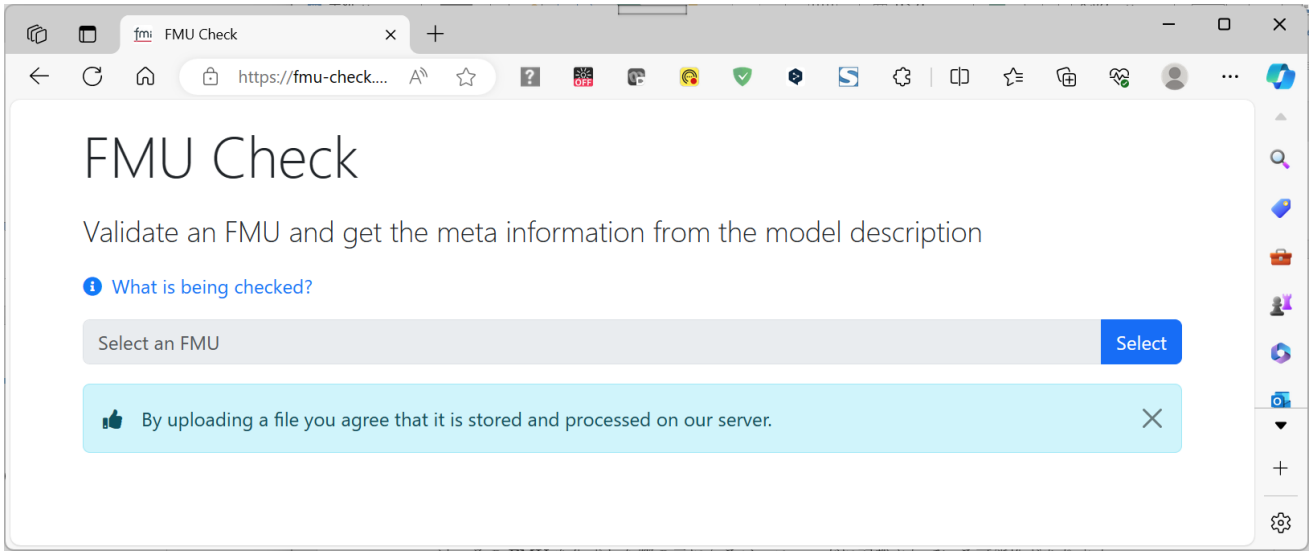


図 4.3.2　Compliance Checker の実行画面

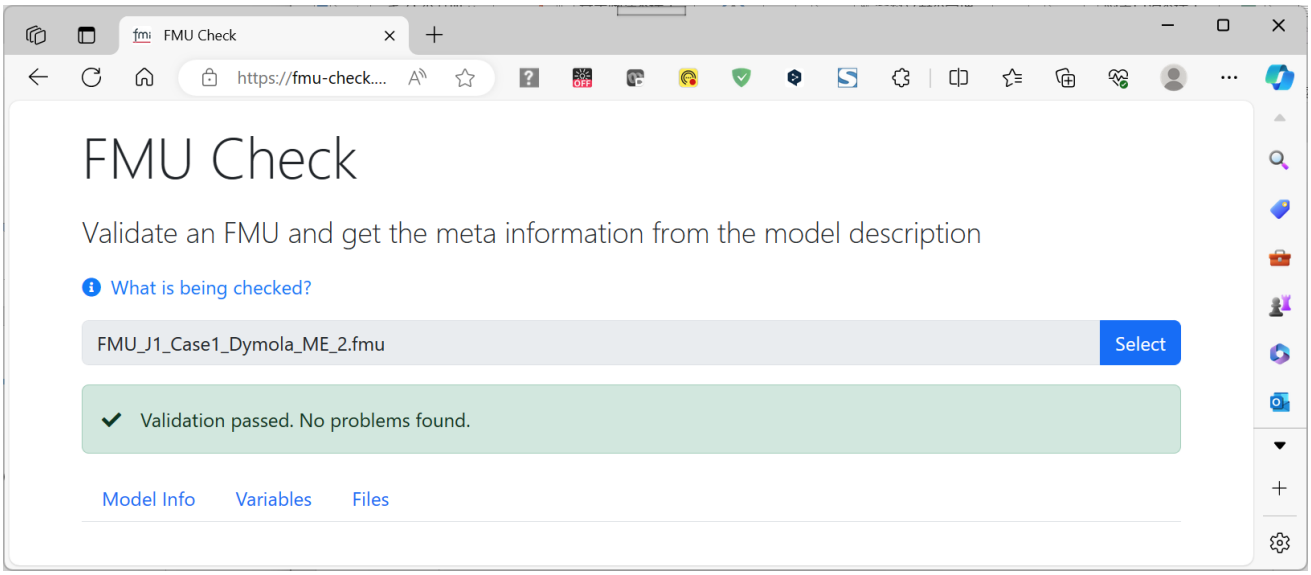


図 4.3.3　Compliance Checker の実行結果画面

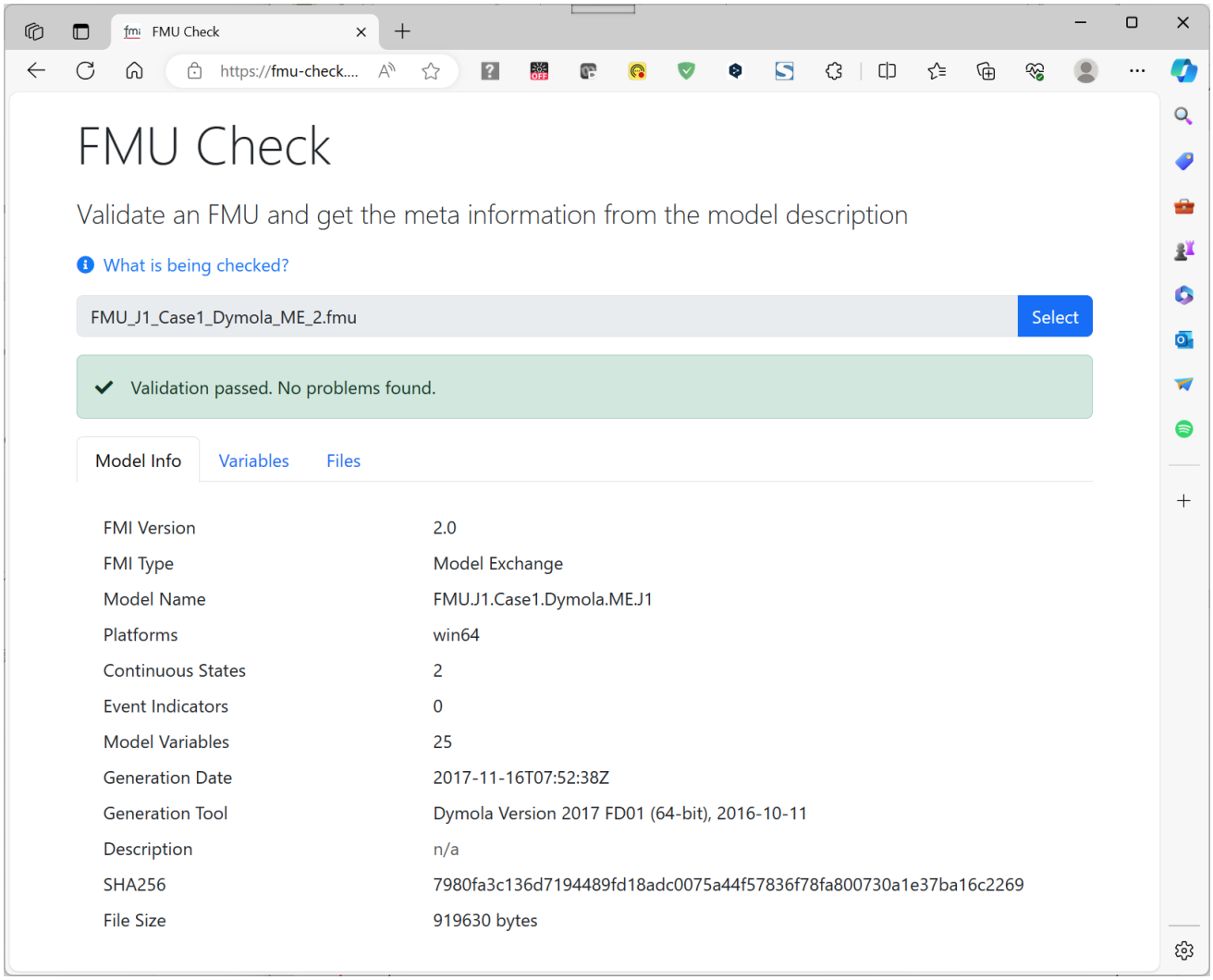
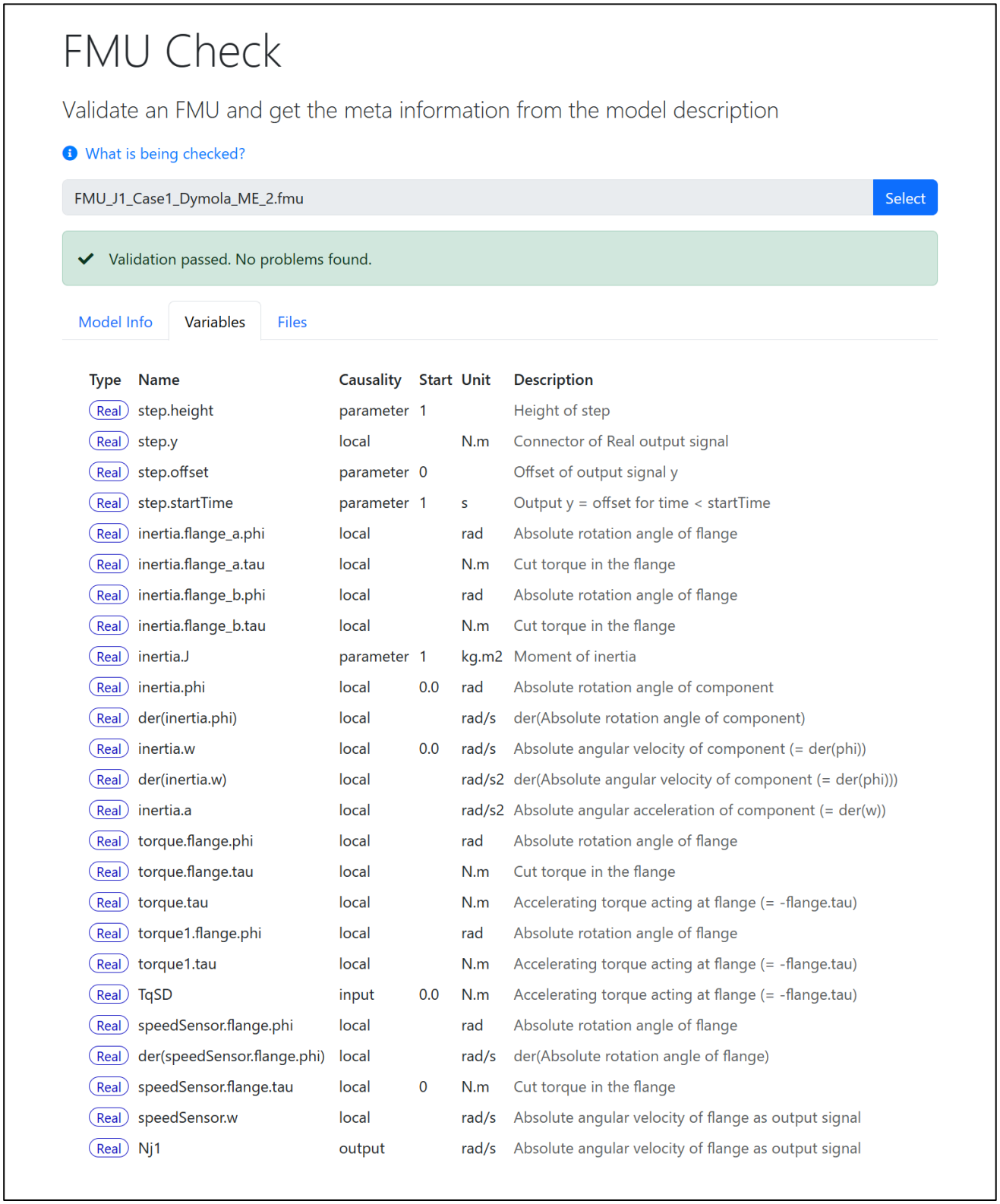


図 4.3.4　Compliance Checker の Model Info の画面



Type	Name	Causality	Start	Unit	Description
Real	step.height	parameter	1		Height of step
Real	step.y	local		N.m	Connector of Real output signal
Real	step.offset	parameter	0		Offset of output signal y
Real	step.startTime	parameter	1	s	Output y = offset for time < startTime
Real	inertia.flange_a.phi	local		rad	Absolute rotation angle of flange
Real	inertia.flange_a.tau	local		N.m	Cut torque in the flange
Real	inertia.flange_b.phi	local		rad	Absolute rotation angle of flange
Real	inertia.flange_b.tau	local		N.m	Cut torque in the flange
Real	inertia.J	parameter	1	kg.m2	Moment of inertia
Real	inertia.phi	local	0.0	rad	Absolute rotation angle of component
Real	der(inertia.phi)	local		rad/s	der(Absolute rotation angle of component)
Real	inertia.w	local	0.0	rad/s	Absolute angular velocity of component (= der(phi))
Real	der(inertia.w)	local		rad/s2	der(Absolute angular velocity of component (= der(phi)))
Real	inertia.a	local		rad/s2	Absolute angular acceleration of component (= der(w))
Real	torque.flange.phi	local		rad	Absolute rotation angle of flange
Real	torque.flange.tau	local		N.m	Cut torque in the flange
Real	torque.tau	local		N.m	Accelerating torque acting at flange (= -flange.tau)
Real	torque1.flange.phi	local		rad	Absolute rotation angle of flange
Real	torque1.tau	local		N.m	Accelerating torque acting at flange (= -flange.tau)
Real	TqSD	input	0.0	N.m	Accelerating torque acting at flange (= -flange.tau)
Real	speedSensor.flange.phi	local		rad	Absolute rotation angle of flange
Real	der(speedSensor.flange.phi)	local		rad/s	der(Absolute rotation angle of flange)
Real	speedSensor.flange.tau	local	0	N.m	Cut torque in the flange
Real	speedSensor.w	local		rad/s	Absolute angular velocity of flange as output signal
Real	Nj1	output		rad/s	Absolute angular velocity of flange as output signal

図 4.3.5　Compliance Checker の Variables の画面

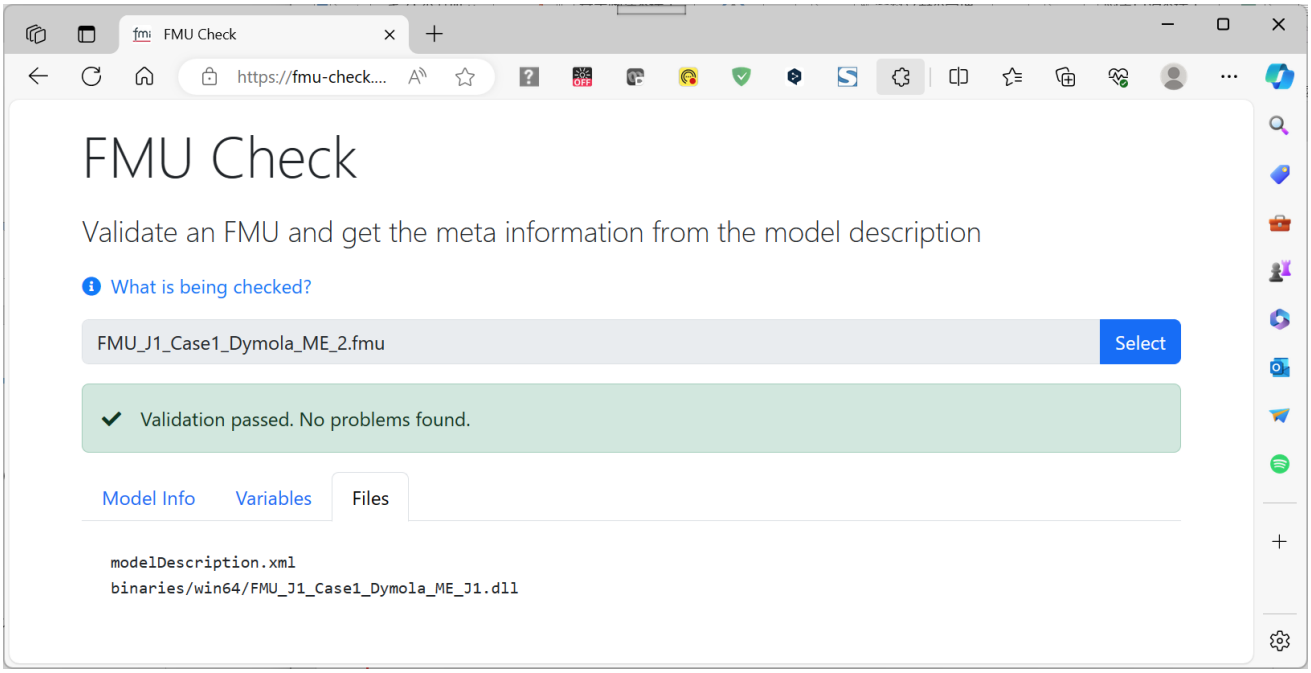


図 4.3.6 Compliance Checker の Files の画面

4.3.2. 取り込みエラー【FMI】

FMI 特有の問題です。FMU を取り込むツールで FMU を取り込もうとする時に、エラーが発生する場合があります。この場合 4.3.1 の Compliance Checker にてチェックを行うことが推奨されます。2.8.4 で説明した名称の付け方によるエラーの場合も多く存在します。

4.3.3. 接続エラー【FMI】

Model Exchange 接続時に発生しうる問題として、FMU を接続し、入出力の繋がりで循環がある場合、本当は存在しない不必要な代数ループが発生することがあります。その場合、model Description File の Dependency をユーザにて変更し、代数ループの解消を行う事も考えられます。
具体例を見てみましょう。これは、別紙の例題にて、回転慣性側（FMU1）として生成された model Description File 例です。

```
<ModelStructure>
    <Outputs>
        <Unknown index="9"/> ⟶ Output signal (speed)
    </Outputs>
    <Derivatives>
        <Unknown index="2"/> ⟶ Acceleration (derivative of speed)
        <Unknown index="4"/> ⟶ Speed (derivative of displacement)
    </Derivatives>
    <InitialUnknowns>
        <Unknown index="1"/> ⟶ Speed
        <Unknown index="3"/> ⟶ Displacement
        <Unknown index="9"/> ⟶ Output signal (speed)
    </InitialUnknowns>
</ModelStructure>
```

FMU 内部で取り扱われる変数には、index が設定されています。積分計算される状態変数および、その他 FMU 生成元モデル上で使用されている変数に対してこの index が設定されます。このケースでは出力される信号は回転数（index=”9”）で、状態変数（index=”1”）の値を検出したものです。
これら 2 つの変数の依存性を宣言されていれば、取り込み側ツールは、出力信号が積分計算される状態

変数であることがわかり、入出力の関係性が代数方程式では無い事がわかります。しかし、上記の例では依存性が表現されていないので、取り込み側のツールが、入出力関係が代数方程式である可能性があると理解し、同様の理解を FMU2 側に行えば、FMU1 と FMU2 間で代数方程式のループが生じると理解されます。実際には、各 FMU の出力値は入力信号を積分計算された値であり、陽解法ソルバを使って解けるはずなのですが、代数ループを解くことが出来る陰解法ソルバを使用することとなります。

陰解法を用いて解く場合は、一巡伝達関数ゲインが 1 以下であることが求められます。

この例にて代数ループを解消するには、下記のように model Description File を変更することが一つの対策例です。

```
<ModelStructure>
    <Outputs>
        <Unknown index="9" dependencies = "1"/>
    </Outputs>
    <Derivatives>
        <Unknown index="2"/>
        <Unknown index="4"/>
    </Derivatives>
    <InitialUnknowns>
        <Unknown index="1"/>
        <Unknown index="3"/>
        <Unknown index="9"/>
    </InitialUnknowns>
</ModelStructure>
```

“dependencies”を宣言することで、出力値（index=”9”）が積分して解かれる状態変数であることが理解でき、不必要な代数ループの発生を防ぐことが出来ます。

上記の例のような対策が必要かどうか、もしくは代数ループ発生時の解消対策可能かどうかは、ツールに依存することをご留意ください。代数ループに関する問題が起きた際の対策の一例としてご紹介しています。

4.3.4.　初期値エラー【共通】

初期値エラーは 2 種類に分類できます。いずれも FMI 特有の問題ではありません

4.3.4.1.　初期値不整合

2 つの FMU 同士、もしくは FMU と取り込みツール側のモデルの双方で同じ変数に対して初期値が規定されていて、その値が異なる場合のエラーです。対処方法は、どちらかの初期値の規定を解除することです。

4.3.4.2.　初期値算出エラー

初期値を明示的に与えていない場合、シミュレーションを実行するツールの中で初期値推定を行うものがあります。その初期値推定ができない場合に出るエラーで、明示的に初期値を与えることで解決します。FMI では FMU 生成時に初期値を明示的に与えることができますが、前述の「初期値不整合」にならないように設定する必要があります。初期値をパラメータとして与えることができる FMU 生成ツールであれば、パラメータとして与えることが望ましいと考えます。

4.3.5. 実行時エラー

4.3.5.1. 初回シミュレーションエラー【FMI】

初期値算出エラーと似ていますが異なるエラーです。FMI の規定では初回シミュレーション時（例えば計算開始時刻を 0 とした場合は time=0 の時）に FMU に入力される値は 0「ゼロ」となります。この入力信号値を用いてシミュレーションしてしまった結果として、ゼロ割によるエラーが起こることが報告されています。

4.3.5.2. 発散【共通】

エラーの代表例を次の 1) 2) に示します。いずれも FMI 特有の問題ではありません。

1) シミュレーションの途中から次第に値が大きくなり、シミュレーションが異常終了する。

ここでは、①モデル単体の数値計算方法に起因するもの、②モデル間の接続方法に起因するもの、の 2 つのケースに分けて説明します。

①モデルは固有値や共振周波数を持ちますが、固有値が使用しているソルバの収束領域に入っているということが、発散しない条件になります。参考までに、Runge-Kutta ソルバの安定領域を図 4.3.7 に示します。複素平面上に表した領域に固有値がある事が収束の条件です。領域は、ソルバタイムステップのΔt によって変化します。Δt を細かくして収束領域の中に収めることや、収束領域の広いソルバを用いることなどの対策が考えられます。

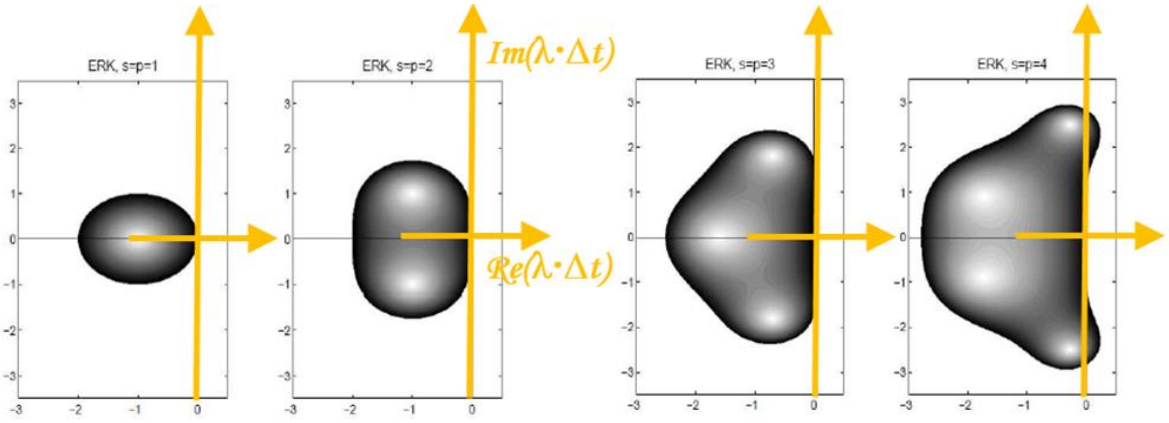
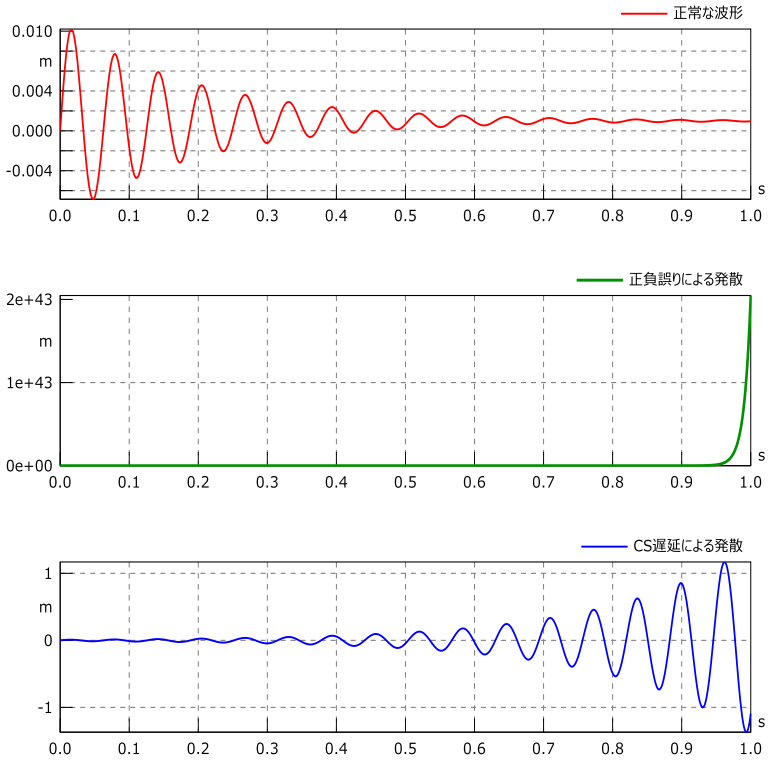


図 4.3.7 Runge-Kutta ソルバの安定領域（1～4 次）

②プラント同士の接続で頻繁に見受けるケースとして、物理量の符号が逆転している場合が考えられます。ネガティブフィードバックであるべきところが、ポジティブフィードバックになってしまっている結果です（図 4.3.8 の中段）。4.1.2 に示したように、エネルギの方向との関係を整理し、生成側ツールで修正後再度 FMU を生成するか、読込側ツールで符号を反転させるかしなければなりません。

2) シミュレーションの途中から信号が振動を起こしシミュレーションが異常終了する。

プラントモデルでの高い固有周波数が影響し、その固有周波数よりも低い周波数の信号授受（による遅延）が影響して不安定になっているケースが考えられます（図 4.3.8 下段）。Co-Simulation の FMU では多くの場合 Communication Step Size を小さくすると回避できます。Model Exchange の FMU では生成環境のソルバで実行すると安定でも、取り込んだ実行環境のソルバでは安定性の違いによりエラーが発生することも多いので、実行ツールのソルバ環境を生成ツール側で理解して、できるだけ種類の近いソルバを用いて実行確認しておくことが望ましいと考えます（3.3.4 参照）。



上段：正常波形

中段：1）物理量の符号誤設定による発散

下段：2）信号授受間隔による発散

図 4.3.8　発散

4.3.5.3. 非収束【共通】

シミュレーションが先に進まず、計算が打ち切られる場合です。

可変ステップソルバを用いたツールで Model Exchange の FMU を使用している場合に発生することが多いエラーです。最小時間刻み幅を小さくするなどの対策を行います。多くの場合現象を不連続にするような事象（イベント）の発生が原因になっていますので、モデル自身に含まれるイベントを起こす部分を見つけ修正する必要があります。プラント同士の接続では、4.2.3 で述べたような「不連続性の高い部位の例」を参照して下さい。また、非因果ツール等で代数拘束条件（代数ループ）を与えた場合、同一時間内での代数方程式をソルバが解こうとするため、条件によっては解が得られずこのような現象が発生します（4.3.1 参照）。

第 5 章　引用文献

[1] 平野 豊：FMI によるモデル流通を目指して，自動車技術会振動騒音部門委員会 No.10-17 シンポジウム,(2017)

[2] 経済産業省 自動車新時代戦略会議： 自動車新時代戦略会議（第 1 回）資料, (2018)
http://www.meti.go.jp/shingikai/mono_info_service/jidosha_shinjidai/pdf/001_01_00.pdf

[3] Blochwitz Torsten, Martin Otter, et al.：The Functional Mockup Interface for Tool independent Exchange of Simulation Models, Preprint of the 8th International Modelica Conference,(2011)
https://ep.liu.se/ecp/063/ecp11063.pdf

[4] 自動車技術会 国際標準記述によるモデル開発･流通検討委員会 モデル接続技術検討 WG： 非因果モデリングツールを用いた FMI モデル接続ガイドライン Ver.1.0, (2015)

[5] Blochwitz Torsten, Martin Otter, et al.：The Functional Mockup Interface for Tool independent Exchange of Simulation Models, Proceedings of the 8th International Modelica Conference,(2011)

[6] Junghanns, T. Blochwitz：10_Years_of_FMI Where are we now? Where we do go? , Keynote speech of the 2nd Japanese Modelica Conference, (2018)
https://www.modelica.org/events/modelica2018japan/presentation/10_Years_of_FMI.pdf

[7] Modelica Association：Functional Mock-up Interface Specification version 3.0

[8] https://fmi-standard.org/docs/3.0/

[9] Yutaka Hirano, Junichi Ichihara, et al.：Toward the actual using FMI in practical use cases in Japanese automotive industry [p195-203], Program of the 2nd Japanese Modelica Conference, (2018)

[10] 日経デジタルヘルスホームページ ：Modelica と FMI―構想設計段階で役立つ CAE 規格（2013)
https://tech.nikkeibp.co.jp/dm/article/COLUMN/20131107/314661/

[11] FMI ホームページ :https://fmi-standard.org/

[12] https://fmi-standard.org/tools/

[13] https://trac.fmi-standard.org/browser/branches/public/Test_FMUs/Compliance-Checker

[14] 経済産業省　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：

[15] 自動車開発におけるプラントモデル I/F ガイドライン（2017）

[16] 緒方 洋介、村上 晋太郎、他：FMI, Model Exchange, Co-simulation におけるシミュレーション安定性に関する考察、自動車技術会 2018 年春季大会 S4-4,(2018)

[17] 市原 純一、斉藤 春樹、他：複数モデリングツールによる FMI を用いた Co-Simulation に関するモデル接続活動紹介（第 2 報）、自動車技術会 2018 年春季大会 S4-2,(2018)

[18] 平野 豊、関末 崇行：FMI によるモデル流通を目指して，自動車技術会 2018 年春季大会 S4-1,(2018)

[19] https://modelica.org/projects.html

[20] https://modelica.org/events.html

[21] https://fmi-standard.org/

[22] https://ssp-standard.org/

[23] https://dcp-standard.org/

[24] https://www.efmi-standard.org/

[25] https://www.prostep.org/en/

[26] https://www.prostep.org/en/projects/smart-systems-engineering/

[27] https://www.prostep.org/fileadmin/downloads/PSI_11_V3_SmartSE_Rec_and_Part_A-I.zip

[28] https://modelica.github.io/fmi-guides/main/fmi-guide/

[29] https://fmi-standard.org/tools/

[30] https://github.com/CATIA-Systems/FMPy

[31] https://pypi.org/project/PyFMI/

[32] Torsten Blochwitz, Andreas Junghanns,Jochen Köhler, Martin Krammer:Overview on FMI, SSP, DCP , 13th International Modelica Conference (2019)
https://fmi-standard.org/assets/literature/FMI_User_Meeting_2019/001_Overview_FMI_SSP_DCP.pdf

[33] FMI ホームページ：https://fmi-standard.org/validation/

[34] MBD 推進センター(JAMBE):自動車開発におけるプラントモデル I/F ガイドライン：
https://www.jambe.jp/system/download