# HLA に基づく製品設計用多分野分散協調シミュレーションの研究

北海道大学大学院工学研究科 ○清水崇文 金井理 岸浪建史

**要 旨**

メカトロ製品等の開発における総合機能検証では、複数分野に跨る協調シミュレーションが有用である。本研究では、HLA に基づく製品設計用の多分野分散協調シミュレーション環境構築のためのモデル統合と実装手法を提案する。

## 1. はじめに

メカトロ製品等は複数分野（機構、電気回路、制御ソフトウェア等）のシステムから構成され、その効率的な開発には、これらの分野で個別に利用されているシミュレータ同士を相互接続した総合機能検証が有用である。従って、このような分散協調シミュレーション用の標準的な接続方法が必要とされている。現在、分散協調シミュレーションの国際標準として、High Level Architecture (HLA)[1]が規定されているが、軍事シミュレータを除くと、HLA 機能内蔵のシミュレータはまだ殆ど無い [2]。製品設計用のシミュレータは、完成ソフトウェアとして提供されるのが一般的であるため、限られた API 等を通じて、ユーザー自身が HLA 機能をシミュレータに追加実装する必要がある[3,4]。

そこで本研究では、機械設計用の 3D-CAD を対象とし、これを HLA 機能付き機構シミュレータへ機能拡張する手法を提案する。本報では、CAD と別分野のシミュレータ間で交換される情報モデルとして HLA で必要とされる FOM の体系的な設計方法、また CAD システム内部と HLA 機能を接続するための Wrapper-Code の体系的な実装方法について提案し、その検証を行う。

## 2. HLA

HLA は分散シミュレーションシステムにおけるシミュレータ間のデータ交換、論理時刻同期を共通化するための接続仕様標準[1]である。図-1 の様に分散シミュレーション全体を Federation、個々のシミュレータを Federate と呼ぶ。各 Federate は HLA とのインターフェースとして Fed.Code、LRC を持ち、通信用ミドルウェアの RTI とメッセージ交換を行う。各 Federate は複数の ObjectClass のインスタンスを持ち、その属性情報は、各 Federate が RTI に対して公開及び購読を宣言することにより、RTI を介して公開側から購読側へ定期的に送信される。また非定期に情報交換を行うには、InteractionClass と呼ばれる通信手段が用いられる。またそれらの Class は Federation Object Model (FOM) と呼ばれるテキストファイルに記述しておき、Federation 起動時に RTI に読み込まれる。

## 3. FOM 設計方法

メカトロニクス製品等の動的挙動のモデルは連続系・離散系の挙動が混在しており、従って CAD と別分野のシミュレータ間で交換される情報モデルを記述する FOM の class は、その両者に対応する必要がある。また、異分野間を跨いで存在している要素（例えばセンサやアクチュエータ）は、分野間で統一した class 定義としなければならない。そこで本研究では、以下の 2 つの FOM 設計方法を提案し、実際にはこれら 2 つの方法を

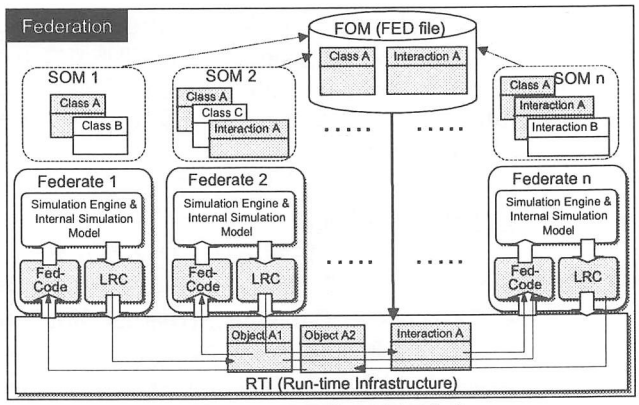



図-1 HLA の基本機能構成

混在して利用することとする。

a) Object 交換方式: この方式では、異なる Federate 間のインターフェースとなるセンサやアクチュエータ等の ObjectClass を、ある Federate 側の視点に基づいて記述する。この方式はシステムの物理構造を反映しやすく、連続状態量の定期的な交換が可能である。

b) Event 交換方式: この方式は、各 Federate 内で不定期に発生する非永続的な事象を InteractionClass として定義する。例えば CAD システム内において、部品間の計測距離を閾値と比較し、この判定結果の変化を近接センサ On/Off という事象を表す InteractionClass として送信することが、この方式に相当する。

## 4. HLA 用 Wrapper-Code 開発方法

既存シミュレータに HLA 機能を追加するには、図-2 の様に、シミュレータ内部モデルから API を通じて FOM で定義された class の情報を読み取り、RTI へこれを送信する LRC 部と、RTI から情報を受信し、これをシミュレータ内部モデルへ API を通じて反映させる Fed.Code 部の両者を、Wrapper-Code として実装する必要がある。これらの実装内容は FOM の class 定義に対応して変化するが、class の変更に対し、コード修正が必要最低限で、体系的に行えることが望ましい。

そこで、本研究では、図-3 の擬似コードのような構造を持つ LRC と Fed.Code の実装方法を提案する。図-3 のコードは Object 交換方式に対する LRC と Fed.Code 例であり、LRC では class*i* の各インスタンスに対し、catchClass*i*Attributes_from_InternalModel() メソッドを呼び出し、送信に必要な属性値を取得した後、RTI のサービス UpdeteAttributeValues()で、これらを RTI へ送信する。一方、Fed.Code 部は、RTI 側からの callback 関数である ReflectAttributeValues()の内部で、受信された Object インスタンスに対し、その class の

applyClass *i* Attribute_to_InternalModel()メソッドを呼び出し、受信された属性値をもとに内部モデルの状態を更新する。ある class の追加削除に対する Wrapper-Code の変更部分は図-3 の点線部分に限定されるため、Wrapper-Code の実装が体系的に行える。Event 交換方式に対しても同様の実装方法を採用できる。

## 5. ケーススタディ

提案手法検証のため、CD-ROM ドライブのトレイ運動制御システムを例とし、HLA に基づく協調シミュレーションを実装した。ドライブの機構シミュレーションは SolidWorks により（以下 CADFed）、制御ソフトウェアは自作 C++プログラム（以下 CntFed）で開発し、これらを RTI（三菱電機 eRTI）を介して協調させた。図-4 にその Federation および FOM の構成を示す。CntFed の制御ロジックは、CADFed 側からのボタン押し下げや、トレイの前後位置を検出する 3 状態スイッチの変化を表した Interaction を入力イベントとし、CADFed 側のトレイ駆動モーターの回転角度を出力として制御する機能をもつ。図-5 はそのシミュレーション結果である。図-5(a) (b)は CntFed が正常に動作している状態である。さらに検証のため、CntFed の状態遷移に意図的な実装誤りを含めて実行した結果、図-5(c)のようにトレイが飛び出してしまう動作が表れ、実装誤りが検証でき、またその制御ロジックの修正の効果も即座に確認できた。

## 6. まとめ

HLA に基づく CAD と別分野を接続した強調シミュレーションで必要とされる FOM の体系的な設計方法、また CAD システム内部と HLA 機能を接続するための Wrapper-Code の体系的実装方法を提案し、ケーススタディによりその有効性を確認した。

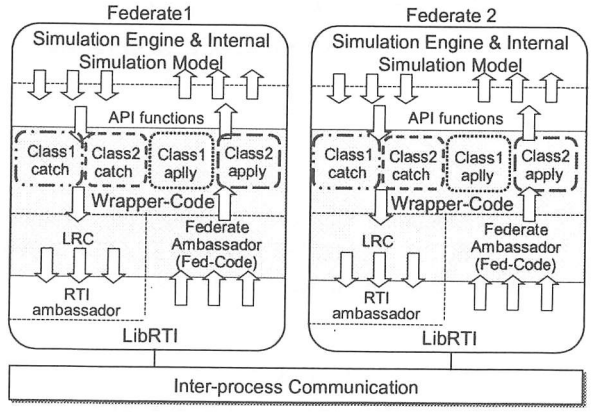
## 参考文献


[1] IEEE Std 1516-2000,M&S High Level Architecture.

[2] Andrei B, et.al. :Distributed simulation of hybrid systems with AnyLogic and HLA, Future Generation Computer Systems,18 (2002) pp.829-839.

[3] S.Pawletta et.al. :HLA- based Simulation within an Interactive Engineering Environment, Proc.4th IEEE DS-RT, (2000)pp.97-102.

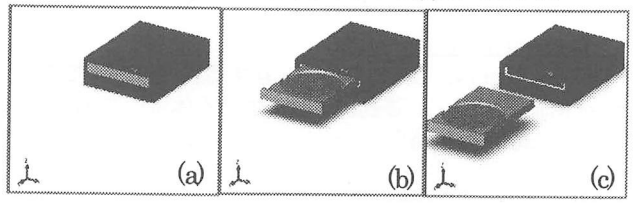
[4]日比野他：生産システム評価のための分散シミュレーションの研究,日本機械学会生産システム部門講演会 2002 講演論文集, (2002) pp. 99-100.

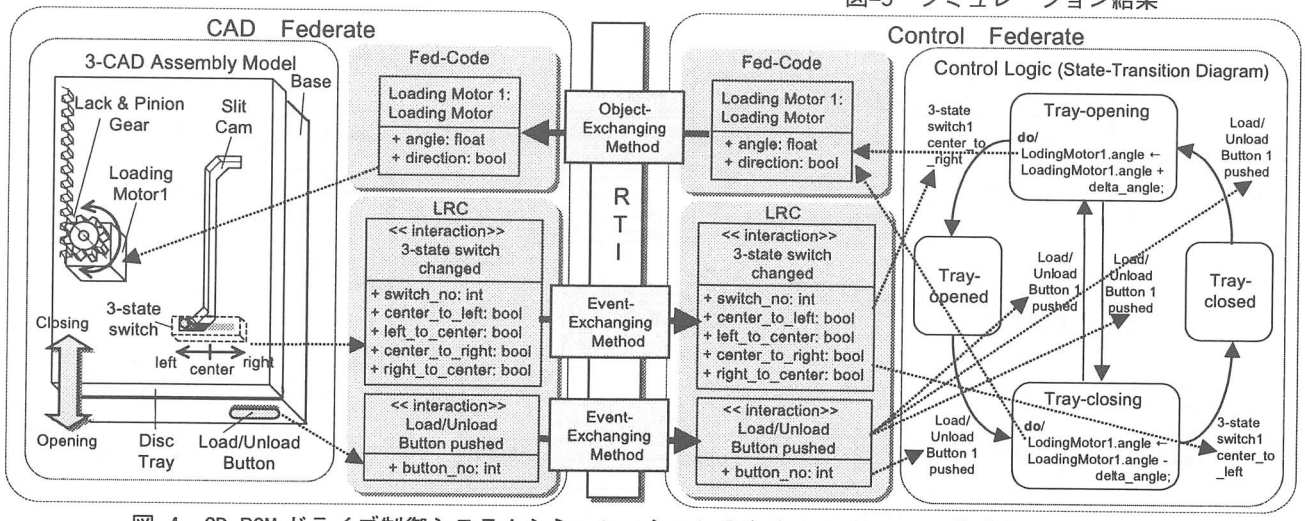

図–2 HLA 用 Wrapper–Code

```
< LRC>
while ( not end of federation execution) {
/* Catch Object Class 1 Attributes */                    Class1用LRC
for j1 to the number of Object Class1 Instances {
Class1_instance j. catchClass1Attributes_from_InternalModel (attrib11, attrib12,...);
RTIAmbassador.UpdateAttributeValues(
  Class1_instance j_designator, attrib11, attrib12,...);
}
/* Catch Object Class 2 Attributes */                    Class2用LRC
for j1 to the number of Object Class1 Instances {
Class2_instance j. catchClass1Attributes_from_InternalModel (attrib21, attrib12,...);
RTIAmbassador.UpdateAttributeValues(
  Class1_instance j_designator, attrib21, attrib22,...);
}
/* Catch Object Class n Attributes */
/* Simulation Time Advancing Codes */
}
< Fed-Code >
virtual void FedAmbasssador::ReflectAttributeValues(
Object_instance_designator, parm i1, parm i2,.... )
{
  decode_object_designator( Object_instance_designator,
received_object_class, received_instance);
switch(received_object_class) {                          Class1用Fed.Code
  case Class1:
received_instance->applyClass1Attributes_to_InternalModel(parm i1, parm i2,...);
break;
  case Class2:
received_instance->applyClass 2 Attributes_to_InternalModel(parm i1, parm i2,...);
break;                                                   Class2用Fed.Code
  default: do nothing;
}
return();
}
```

図–3 Object 交換方式での Wapper–Code 例




図–5 シミュレーション結果




図–4 CD–ROM ドライブ制御システムシミュレーションのための Federation 構成と FOM モデル