

アプリケーション非依存型の製造ステーションモデル構成法

— UML に基づくステーション参照モデル記述法とロボットシミュレータへの応用 —

旭川工業高等専門学校
北海道大学工学部

○戸村 豊明
金井 理, 岸浪 建史

要 旨

生産セルを構成するステーションのモデリングは、そのモデルのアプリケーションごとに別個に行われており、現状では、アプリケーションに依存しないステーション固有の静的構造、状態遷移をアプリケーション間で共有可能なモデリング手法は提案されていない。本報では、オブジェクト指向モデリング言語 UML (Unified Modeling Language) で記述されたステーション固有の静的構造、状態遷移からなるステーション参照モデルを、実機ステーションの仕様を基にサブクラス化するモデリングプロセスを提案する。その実例として、多軸ロボットステーションのモデリングを行い、これをロボットシミュレータ内へ実装する事で、参照モデルとモデリングプロセスの有効性を明らかにする。

1. はじめに

ステーションモデルのアプリケーションとして、シミュレーションとプログラミングを考えた時、その入力と出力であるプログラムとデバイスのアクティビティは、両アプリケーション間で逆転の関係にあると考えられる。また、両アプリケーションでそれぞれ利用されるステーションモデルは、アプリケーション非依存の共通利用部分として固有の静的構造、状態遷移を持つと考えられる。しかし、現状では、これらの部分をアプリケーション間で共有可能とするモデリング手法は提案されていない。

本報では、UML^[1]で記述された固有の静的構造と状態遷移からなるステーション参照モデルを、実機ステーションの仕様を基にサブクラス化するステーションモデリングプロセスを提案する。その実例として、本報では、多軸ロボットステーションをモデリングし、それをロボットシミュレータ内へ実装する事で、提案する参照モデルとモデリングプロセスの有効性を明らかにする。

2. ステーションモデルとシミュレータ

ステーションのシミュレータは、一般的に図1のような構造を持つ。ステーションモデルは、コントローラ、デバイス、プログラム、アクティビティのオブジェクトからなる。コントローラがプログラムを実行する事で、デバイスとの相互作用が起こり、アクティビティが生成・可視化され、それがアプリケーションの出力となる。

3. ステーション参照モデル

本研究で提案するステーションモデリングプロセスは、図2のように、分析、設計、実装から構成される。ステーションモデルおよび参照モデルの記述に用いられる

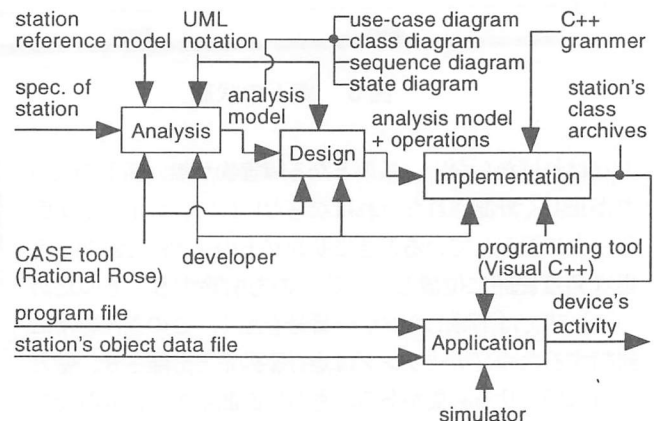


図2. 本研究で提案するステーションモデリングプロセス

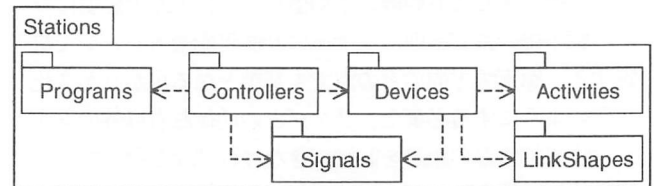


図3. ステーション参照モデルのパッケージ構成

UMLは、他の言語に比べ、分析・設計でのダイアグラム間の整合性に優れているので、ダイアグラムの作成と詳細化のみで、モデルの大部分の実装が可能である。

ステーション参照モデルは、クラス図^[1]、各クラスの状態図^[1]からなり、分析においてサブクラス化される。参照モデルにおけるコントローラに対する要求条件は、実機固有の命令セットにおける各命令に対応する振る舞いを記述できるよう、CPU、メモリ上のデータ構造を容易にサブクラス化できる事である。一方、デバイスに対する要求条件は、実機固有のサーボ機構を記述できるよう、アクチュエータとドライバを容易にサブクラス化できる事である。よって、ステーション参照モデルでは、これらサブクラスに対するスーパークラスが定義されている。

図3はステーション参照モデルのパッケージ^[4]構成を示しており、各パッケージは、それぞれ以下のクラス群を定義するためのモジュールである。

- Controllers: Controller クラスを定義 (図4)。
- Devices: Device クラスを定義. アクチュエータが持つ能動的ジョイントの作動により、受動的ジョイントで結合されたリンク間の相対運動が生まれる。
- Activities: 相対運動の順序 (Activity クラス) を定義。

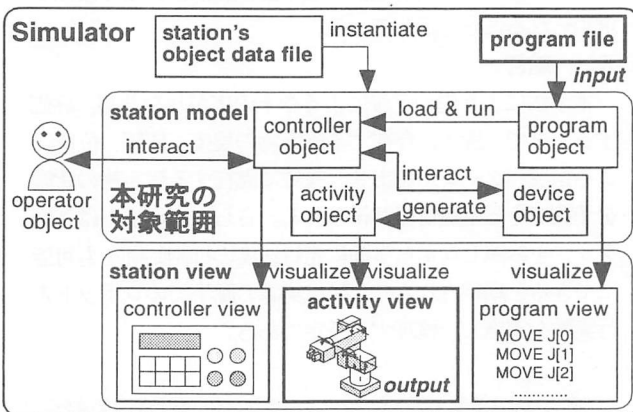


図1. ステーションのシミュレータの構造

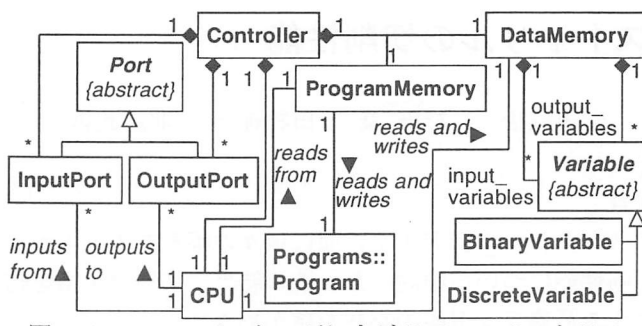


図 4. Controller パッケージにおける Controller クラス

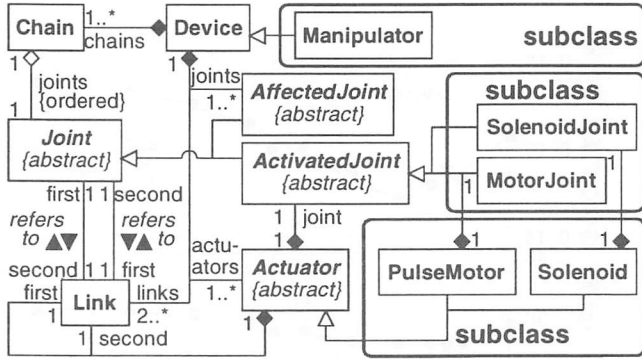


図 5. Device クラスのサブクラス化

4. 多関節ロボットステーションのモデリング

本報では、ステーションのモデリングの具体例として、4 軸ロボットステーションを取り上げる。このロボットは、4 つのパルスモータ（関節用）とソレノイドバルブ（ハンド用）で駆動されるものとする。

ステーションモデリングプロセスにおける分析の手順は、以下ようになる。

(1) ユースケースのモデリング

コントローラをシステム、プログラムをアクタ¹¹⁾（システムと相互作用するオブジェクト）とみなし、コントローラの仕様で定められた命令セットを基に、命令の分類を行う。本報の例では、単軸動作、全軸動作、ハンド動作、待機、プログラム終了といった分類が得られた。同類の命令間で共通の原子的機能、および同類の各命令の実行をユースケース¹²⁾として定義する事で、ユースケース¹³⁾を作成する。

(2) Controller クラス、Device クラスのサブクラス化

実機ステーションの仕様を参照する事で、サブクラス化が行われる。図 4 の CPU クラスは命令の解釈とその振る舞いが実機固有なので、本報の例ではサブクラス **RobotCPU** が定義され、また、**DataMemor** クラスがジョイント変数 (**JointVariable** クラス) を記憶する為にサブクラス **RobotDataMemory** が定義される。また、図 5 のように、Device クラスの部分クラスに対してもサブクラスが定義される。

(3) ユースケースにおけるオブジェクト間相互作用の定義

(1) で得られた各原子的機能において、関係するオブジェクトを特定し、オブジェクト間のメッセージ交換のシーケンスを定義する事で、図 6 に示すようなシーケンス図が得られる。RobotCPU オブジェクトは自身に **controlJoint()** メッセージを送信し、モータの回転方向を出力した後、指令値を出力する。また、各命令の実行は、原子的機能の組み合わせとして表現される。

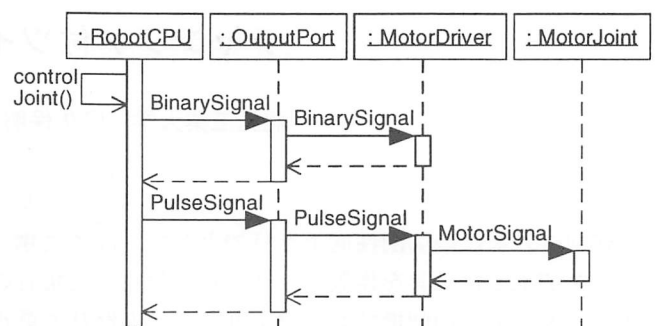


図 6. ユースケース「Controlling a joint」のシーケンス図

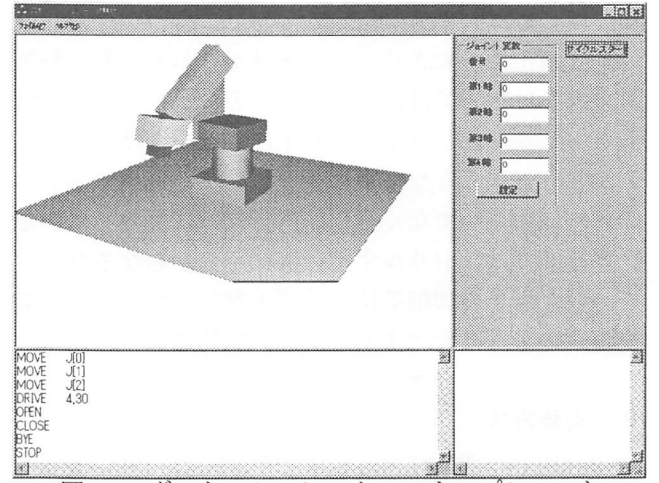


図 7. ロボットシミュレータのスナップショット

(4) サブクラスの状態遷移の定義

(3) で得られた各オブジェクトが受信するメッセージをイベントとする状態図を各サブクラスごとに作成する事で、各サブクラスの状態遷移を定義する。設計の手順は、以下ようになる。

(5) サブクラスへの操作の追加

(4) で得られた各イベントを、操作としてクラス図における各サブクラスへ追加する。

(6) 操作で実行される手続き (メソッド) の設計

アクティビティ¹⁴⁾、擬似言語等を用いて、(5) で得られた各操作において実行される手続きを設計する。

Rational Rose を用いて、(1)~(6) により得られた UML 記述のステーションモデルは、C++ により実装可能である。

5. ロボットステーションモデルのシミュレータへの実装

ステーション参照モデルおよびモデリングプロセスの有効性を明らかにするために、本報では、IBM-PC 上で実行可能なロボットシミュレータのプロトタイプを Visual C++ で開発した。その実行画面を図 7 に示す。サイクルスタートボタンを押すとプログラムが実行され、ロボットの動作がデバイスのアクティビティとして得られた。

6. まとめおよび今後の課題

本報では、ステーション固有の静的構造、状態遷移からなる UML 記述のステーション参照モデルを、実機ステーションの仕様を基にサブクラス化するモデリングプロセスを提案した。また、多軸ロボットステーションをモデリングし、ロボットシミュレータ内へ実装する事で、参照モデルとモデリングプロセスの有効性を明らかにした。

今後は、多関節ロボットステーションモデルのプログラミングへの利用について議論する予定である。

参考文献
[1] H. E. Eriksson 他 : UML ガイドブック, トッパン, 1998.