並行システムにおけるアーキテクチャ記述変換ツールに関する研究

2009SE144 草川紘平　2010SE023 藤原光翼　2011SE229 佐藤大輔
指導教員：張漢明

1　はじめに

状態遷移機械の集合であらわされる並行ソフトウェア設計の検査に用いる主な検査方法には，モデル検査がある．ソフトウェア開発において設計を UML[7] でおこなうのが一般的になっているが，モデル検査をおこなうためにはモデル検査で用いる検証用コードで記述しなおす必要がある．本研究室では，アーキテクチャを UML で図式表現し，UML から検証用コードに自動変換するアーキテクチャ記述変換ツールを用いることにより並行システムの検証をおこなう方法が提案されている [8]．しかし，検証用コードには様々な表記法が存在するため，表記法の変更や検証のための新たな構成要素を作成する必要に迫られることがある．そうなった場合，現在の変換ツールはモデルの変更があった際，プログラムの書き換えに手間がかかるという問題がある．

本研究の目的は，記述モデルの変更に柔軟なアーキテクチャ記述変換ツールの設計と試作である．図式表現の表記法の書き換え，生成する詳細なコードの表記法の変更，複数の言語に柔軟に対応するツールの作成をおこなうことにより，モデルの変更があった場合のツール改良の効率化を目指す．

本研究の基本的なアイデアは，MDA[3] に基づくアーキテクチャ記述モデルを構築することである．MDA とは，ソフトウェア開発手法であるモデル駆動アーキテクチャのことである．MDA に基づいて言語の性質に非依存なモデルである PIM と言語の性質に依存したモデルである PSM の設計をおこなう．PIM と PSM の対応関係を定義し，PSM と具体的な構文の対応関係を定義して，モデル間の変換を可能にする．

成果として，PIM と PSM を作成し，ツールを作成した．本稿では PIM をアーキテクチャの構成要素とし，PSM を UML，CSP[1]，Promela[6] としたモデルを構築した．そして，PIM と PSM 間の変換系の作成と，PSM から具体的な構文を生成する変換系の作成をおこなうことによって，PSM 間の変換が可能なツールの概要を示す．

2　基本的なアイデア

2.1　要素技術

2.1.1　並行システムのアーキテクチャ記述

本研究では並行システムのアーキテクチャは，複数の並行に動作する状態遷移機械 (ConcurrentStateTransitionMachine，以下 CSTM) の集合で表現する．CSTM は受信したイベントに応じて状態を遷移し，状態遷移時にアクションとして CSTM の処理を実行する．状態遷移時にほかの CSTM にイベントを送信する．この各 CSTM の協調動作によって，並行システムの機能を実現する．並行システムのアーキテクチャ記述は UML 記述を用いる．キューを介したイベント送受信を図 1 に示す．

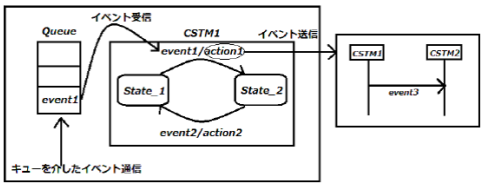

図 1　キューを介したイベント送受信

2.1.2　並行システムの検証

本研究室で研究されている並行システムの検証は，システムの振舞いの検証とコンポーネントの状態の一貫性の検証がある．モデル検査器を用いてそれらを検証し，また性質に合わせ，検査器を使いわける必要がある．本研究室では振舞いの検証には FDR[4]，状態の一貫性には SPIN[5] の検査器を使い，それぞれの検証用コードは CSP と Promela が用いられる．

2.2　モデル駆動アーキテクチャ（MDA）

モデル駆動アーキテクチャ（以下，MDA）は Object-ManagementGroup が提案する，モデルを中心としたソフトウェア開発の枠組である．MDA が示す開発プロセスではモデル開発を適切な言語を用いて定義することにより，ソフトウェア開発における，モデル，ソフトウェアコンポーネントの再利用を図っている．

MDA が示す開発プロセスでは，MDA の基盤ともよべる MOF[3] により，言語の構成要素を記述するための一貫した方式を定義する．統一まずはじめにプラットフォーム独立モデル（PIM）を定義する．PIM は名のとおりプラットフォームに依存しないモデルであり，PIM の設計では開発の対象となるソフトウェアにおける，プラットフォームに特化しないソフトウェアの構造と，ソフトウェアコンポーネント間の相互関係を定義する．適切に定義された PIM は，対象となるプラットフォームへの変換規則を定義することにより，プラットフォーム特化モデル（PSM）に変換される．PSM はプラットフォームに依存したモデルであり，PIM から PSM への変換は自動化することが可能である．

2.3　MDA に基づいたアーキテクチャ記述モデル

ツールの作成にあたり，モデルとして表現される構造的仕様のガイドラインを提供するソフトウェア開発手法であ

る MDA に基づくモデルの構築をおこなう．言語の性質に非依存なモデルである PIM と言語の性質に依存したモデルである PSM を設計する．PIM と PSM の対応関係を定義し，PSM と具体的な構文の対応関係を定義し，定義した対応関係を元にプログラミングをおこないツールの作成を可能とする．本研究において具体的な構文は，UML，CSP，Promela とする．MDA に基づくモデルのアイデアを図 2 に示す．

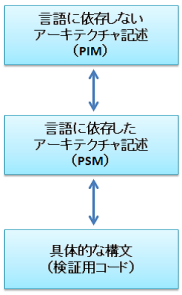

図 2　MDA に基づく本研究のアイデア

3　MDA に基づくモデルの構築

3.1　アーキテクチャ記述の概要

アーキテクチャ記述をおこなうために必要な構成要素を定義する．

◆構成要素

- オブジェクトモデル
  - -概念クラス
  - -クラス間の関連
- インスタンスモデル
  - -インスタンス
  - -インスタンス間の関連
  - -概念クラス
- コンポーネント
  - -階層構造
  - -状態遷移機械
    - ・状態
      - -初期状態
      - -終了状態
    - ・遷移
      - -イベント
      - -アクション
  - -アクション
    - ・イベント送信
      - -送信元，送信先コンポーネント
      - -イベント

これらの構成要素を元に，各要素の対応関係を表現するためにオブジェクトモデルを構築する．

3.2　アーキテクチャ記述モデル

検証するアーキテクチャの構成要素から PIM のモデルを作成する．

◆アーキテクチャ記述のモデル化

一例として，コンポーネントを挙げる．

- **コンポーネント**

  コンポーネントは階層構造を持ち，状態遷移機械とアクションで構成されている．特に，言語に依存したモデルでは状態遷移機械とアクションの記述方法に違いがある．コンポーネントの階層構造を図 3，状態遷移機械の構造を図 4，アクションの構造を図 5 に示す．

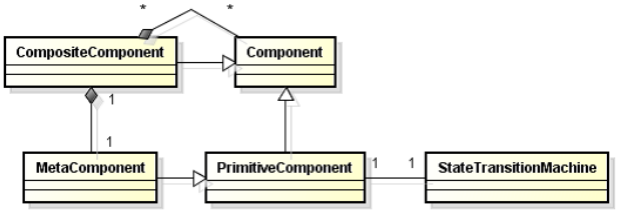

図 3　コンポーネントの階層構造

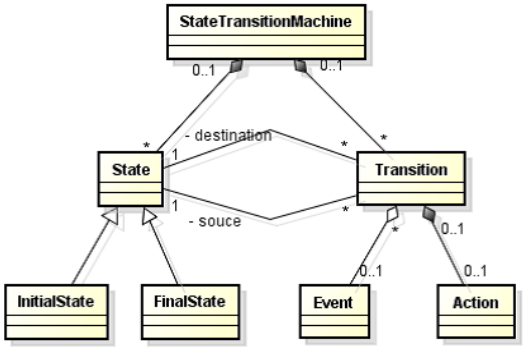

図 4　状態遷移機械の構造

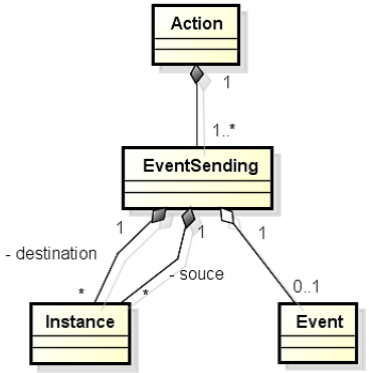

図 5　アクションの構造

3.3 言語に依存した記述モデル

言語に依存したアーキテクチャ記述モデルを定義する．検証用コードを生成するにあたり，UML，CSP，Promela のアーキテクチャ記述モデルを作成する．アーキテクチャ記述には状態遷移機械とアクションを定義する．

3.3.1 UML によるアーキテクチャ記述

アーキテクチャ記述を検証する際に，Astah[2] を用いて構造を図式表現する．システムの構造はオブジェクトモデルとインスタンスモデルで表現し，各コンポーネントはそれぞれ，状態遷移機械はステートマシン図，コンポーネントの振舞いはシーケンスで表現する．PIM の各モデルと対応させるために，各図を UML のメタモデルに基いて，表現に必要な要素だけを取り出してモデル化する．一例として，シーケンス図を図 6 に示す．

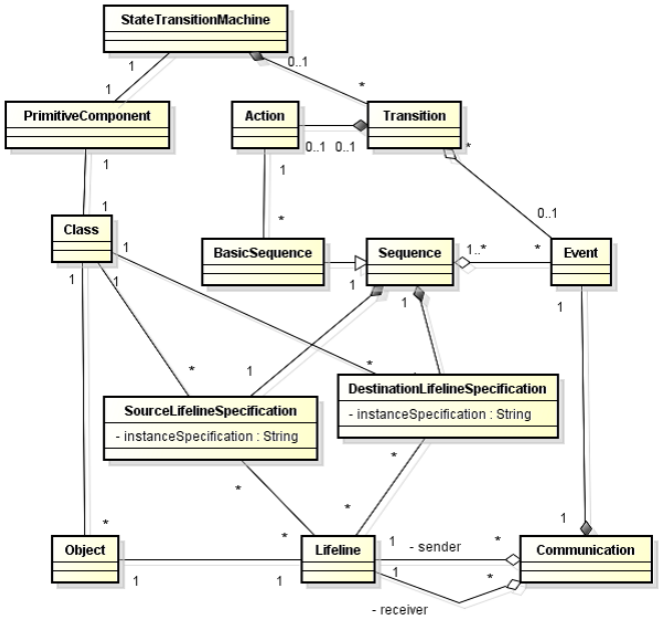

図 6 シーケンス図のメタモデル

3.3.2 CSP によるアーキテクチャ記述

CSP でアーキテクチャ記述を表現するには，状態遷移機械とアクションを用いる．CSP の構成要素は，ライブラリに依存する．CSP のモデルの一例として，CSP における状態遷移機械を図 7 に示す．

3.3.3 Promela によるアーキテクチャ記述

Promela でアーキテクチャ記述を表現するには，状態遷移機械とアクションを用いる．Promela のモデルの一例として，Promela における状態遷移機械を図 8 に示す．

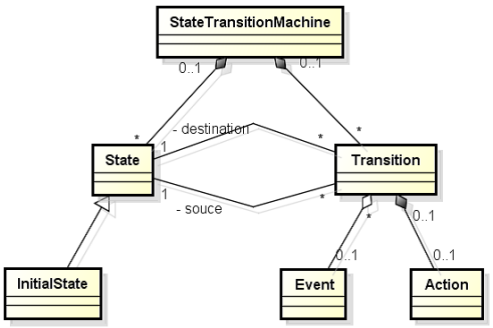

図 7 CSP における状態遷移機械の構造

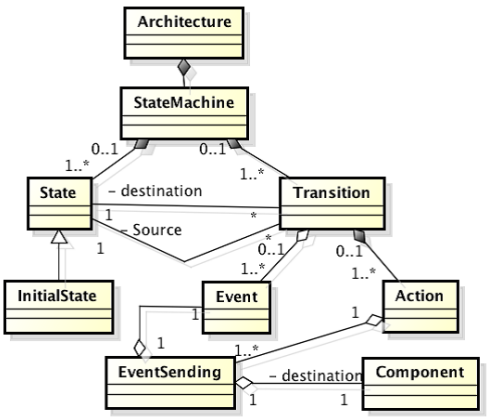

図 8 Promela における状態遷移機械の構造

4 PIM と PSM 間の変換系の構築

PIM と PSM の変換系を構築するために，構築したモデル同士の構成要素の対応関係を定義する必要性がある．対応関係を定義することにより，モデル間の変換が可能になる．

4.1 UML から PIM への変換

Astah を用いて作成された UML の情報は, UML のアーキテクチャ記述モデルの構造に格納する．UML のデータの構造と PIM との対応関係を定義することにより，PIM への変換ができる．対応関係の一例として，シーケンス図の UML とアクションの PIM との対応関係を図 9 に示す．

4.2 PIM から CSP への変換

PIM と作成した CSP のアーキテクチャ記述モデルの対応関係を定義することにより変換をおこなう．CSP モデルの構造になっている情報から実際の CSP の構文を生成する．対応関係の一例として，状態遷移機械の PIM と CSP との対応関係を図**??**に示す．

4.3 PIM から Promela への変換

CSP と同様に作成した Promela のモデルとの対応関係を定義，その構造の情報から構文を生成する．対応関係の一例として，PIM と Promela の対応関係を図 11 に示す．

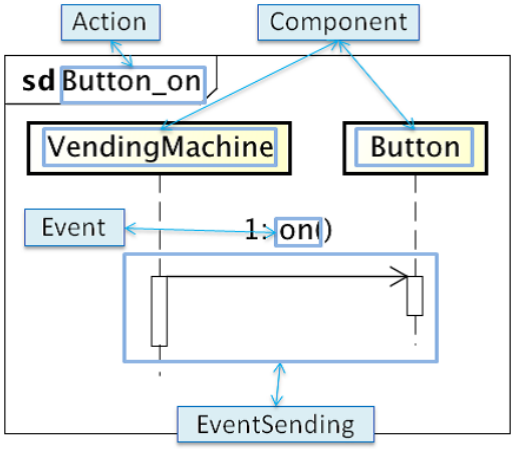

図 9　シーケンス図における UML と PIM の対応関係

PIM	PSM
StateTransitionMachine	StateMachineMachine
State	State
InitalState	InitalState
Event	Event
Action	Action

図 10　PIM と CSP の対応関係

PIM	PSM
InstanceModel	Architecture
StateTransitionMachine	StateMachine
Transition	Transition
EventSending	EventSending
State	State
Event	Event
Action	Action

図 11　PIM と Promela の対応関係

5　考察

5.1　変更の容易性

本研究では，MDA に基づいてモデルを定義した．変更の容易性として二つの点を挙げる．

5.1.1　図式表現の変更

例として UML 記述におけるステレオタイプを挙げる．ステレオタイプを用いてイベントの送信，受理を表現している．ステレオタイプ名の変更したい場合に，UML と PIM との変換系を作成しなおすだけでツールの作成を可能とすることができる構造となっている．

5.1.2　新たな言語の追加におけるツール作成

MDA に基づいてツールを作成したことにより，PSM から新たな言語の PIM の対応関係を定義することで新しい言語に対応したツールの作成を容易におこなうことができるソフトウェアの構造となっている．

5.2　相互変換

PSM と PIM の相互的な対応関係を定義しているので，PIM から PIM への変換が可能である．つまり，検証用コードから図式表現である UML 記述への生成も可能とできる構造となっている．

6　終わりに

本研究では，柔軟なアーキテクチャ記述変換ツールの設計と試作をおこなった．今後の課題として，CSP 記述を用いたモデル検査では環境と仕様の記述が必要となるので，それらを UML 記述から自動生成できるようにツールを改良する点が挙げられる．この点を改善することで自動生成された CSP 記述を用いて検証をおこなうことができる．

参考文献

[1] C. A. R. Hoare, Communicating Sequential Pro-cess, Prentice-Hall, 1985.

[2] Change Vision, Inc., “ astah* professional — UML モデリングツール — Astah, ” http://astah.change-vision.com/ja/product/astah-professional.html, 2009.

[3] David S. Frankel, “ MDA モデル駆動アーキテクチャ, ” 株式会社エスアイビー・アクセス, 2003.

[4] Formal Systems(Europe), “ Formal Systems（Europe）Ltd, ” http://www.fsel.com/, 2010.

[5] G. J. Holzmann, “ The Model Checker Spin, ” IEEE TRANSACTIONS ON SOFTWARE ENGINEER-ING, vol. 23, no. 5, 1997.

[6] Holzmann, G. J, “ The SPIN Model Checker: Primer and Reference Manual, ” Addison-Wesley, 2004.

[7] Object Management Group , “ Unified Modeling Langage(UML), ” http://www.uml.org/, 2008.

[8] 小栗達也，山内宏也，“ アーキテクチャ記述の振舞い検証支援ツールに関する研究, ” 南山大学 2010 年度卒業論文, 2010.