役割指向による並列分散システムの記述方法に関する研究

KASHIWAGI, Takahito / 柏木, 孝仁

(出版者 / Publisher)
法政大学大学院情報科学研究科
(雑誌名 / Journal or Publication Title)
法政大学大学院紀要. 情報科学研究科編
(巻 / Volume)
8
(開始ページ / Start Page)
141
(終了ページ / End Page)
144
(発行年 / Year)
2013-03
(URL)
https://doi.org/10.15002/00009856

役割指向による並列分散システムの記述方法に関する研究

Study of a Method for Describing Parallel and Distributed Systems with Role-Orientation

柏木　孝仁
Takahito Kashiwagi
法政大学大学院情報科学研究科情報科学専攻
E-mail: takahito.kashiwagi.6k@stu.hosei.ac.jp

Abstract

Role-orientation is one approach for building Agent-Based Models (ABMs). Role-orientation provides a method of constructing ABMs with concept of "agent" and "stage", "role". We assume that role-orientation has a property for building parallel systems. For example, "agent" of role-orientation's concept performs parallel processing, and "stage" of role-orientation's concept synchronizes of parallel processing. We propose a method for describing parallel and distributed systems with role-orientation. Parallel and distributed systems consist of one client and servers. One aspect of major issues in constructing parallel and distributed systems is the difficulty of implementing parallel processes and network processes. We try to solve the problem in constructing parallel and distributed systems through abstracting the systems using concept of role-orientation. In this research (1) we design parallel and distributed systems with role-orientation and (2) we implement designed systems in agent-system, (3) we describe implemented systems using our role-oriented template generator that is a support tool for describing role-orientation. By using our method, we can reuse implementation and role-oriented description of parallel and distributed systems focusing on stage and role. This property of role-orientation is expected to work effectively on developing customizable parallel and distributed systems.

1. はじめに

役割指向は，Agent-Based Models（ABMs）を構築するための一つの手法として提案された概念である．ABMsは，エージェントを用いて実世界の現象を再現する．役割指向における「役割」および順序制約を表す「ステージ」の概念は，実世界の現象をエージェントの振る舞いと時系列という2軸に基づいて複数のサブモデルに分割して記述することを可能とし，ABMsにおいては直感的なモデルの構築方法を提供する．本来，役割指向はABMsの構築に用いられる概念であるが，実世界において並列に動作するエージェントをモデル化する性質をもつため，並列システムの記述に応用できると考えられる．すなわち，並列システムを構築するために役割指向がもつ特徴的な概念を用いることによって，エージェントによる並列処理の実行，ステージ概念を用いた並列処理の同期を行う，などのシステムの記述，実装法が可能となると考えられる．

本研究では並列システムの中でも並列分散システムに着目し，役割指向による並列分散システムの有効な記述を行うことを目標とする．並列分散システムは，1つのクライアントとサーバ群からなる構成のシステムである．役割指向による有効な記述とは，役割指向の利点を用いた記述である．すなわち，システムをエージェントやステージを用いて記述できるとともに，具体的なシステムの実装を隠蔽してシステムの設計が行える，という特質が得られることである．なお，本来並列分散システムにおいては，実行速度の効率が求められるはずだが，本研究では役割指向によるプログラムの記述性についてのみ注目する．

2. 背景

役割指向とは，エージェントシミュレーションシステムSOARS（Spot Oriented Agent Role Simulator）[1]において，Agent-Based Models（ABMs）を構築するために提案された概念である．役割指向は，順序制約を表す概念であるステージ，ステージにおける処理の実行を行うエージェント，エージェントが実行する処理をステージと関連付けモジュール化を行う機能を担う役割という3つの概念を持つ．SOARSでは，役割指向の概念である，ステージとエージェント，役割のほかにスポットという概念を中心にシミュレーションモデルを構築する．スポットはエージェントが相互作用する場を表現したオブジェクトである．SOARSにおいて，社会シミュレーションや感染症シミュレーション，経済シミュレーションなど，シミュレーションの開発に役割指向の概念が有効に働くことがわかってきた．しかし，シミュレーションモデルの構築以外に役割指向の有効性を示す事例は少ない．

SOARSにおいて，役割指向のステージと役割が，シミュレーションモデルの構造化に効果的に働くことに着目して，役割指向をソースコードの生成に利用したツールが，役割指向テンプレートジェネレータである[2][3]．本ツールでは，(1)ステージはソースコードの構造（ソースコードの断片の順序）を定義する．(2)エージェントは，ステージに定義された構造に従いソースコードを出力するオブジェクトと捉えられる．(3)役割は，ソースコードの断片をステージと関連付けてまとめる機能としてはたらく．役割指向の実際の記述では，メタ文字を用いて，ステージ（/）やエージェント（@），役割（なし），ソ

Supervisor: Associate Prof. Akira Sasaki

ースコード（タブ）を定義する．メタ文字をディレクティブの1種として用いることによって役割指向を記述でき，ソースコードを構造的に生成する機能から，より簡潔な記述方法を用いて汎用言語によるエージェント実行系の実装を使用することができる[3].

本研究において，役割指向を用いて記述するのは並列分散システムである．記述する並列分散システムは，1つのクライアントとサーバ群からなる構成のシステムであり，今回，特に記述するものはシステムのクライアントである．クライアント側の処理では，サーバとの通信処理があり，また，各サーバへの仕事の依頼を並列に行う必要がある．このように，並列分散システムのクライアント側では，ネットワーク処理や並列処理が必要であり，システム構築は困難である．並列分散システムを構築するための，ライブラリやフレームワークはいくつか存在するが，1度記述したシステムを再利用し，他のシステムへ拡張することが容易であるとは限らない．

3. 役割指向による並列分散システムの記述および実装方法

役割指向を用いて並列分散システムを記述する方法と，記述からシステムの実装を作成する方法を説明する．

並列分散システムのクライアントを記述する方法であるが，クライアントには，サーバとの通信処理や，各サーバへの仕事の依頼を並列に行う処理がある．役割指向の概念であるステージを用いることにより，並列に動作するエージェントを制御することが基本的なアイデアである．より具体的には，処理全体のうち，並列に処理できる部分は，1つのステージとして他の処理と区別する．複数に分割したステージの中で，並列に処理できるステージでは，エージェントは並列に処理を実行する．例えば，1体のエージェントが1つのサーバへの処理（ネットワーク処理や，並列性のある仕事の依頼の処理）を行うとする．サーバへの仕事の依頼の処理は並列に処理することができるため，1つのステージとする．サーバへ仕事を依頼するステージでは，複数のエージェントは並列に処理を実行する，という仕組みである．ただし，並列に処理できる箇所をまとめて1つのステージとしてしまうと，役割指向記述の汎用性が失われてしまう．役割指向記述をカスタマイズしていくことを考慮して，適切にステージを定義する必要がある．

並列分散システムのクライアントを記述するための他のアイデアとして，複数のエージェントとやりとりするオブジェクトをスポット（SOARSの概念）とする．SOARSにおいて，エージェント間では直接通信をせず，エージェントはスポットとのみやりとりをする仕組みにすることにより，システムの記述が複雑になるのを防ぐ．ここではこの手法を応用し，クライアントを1つのスポットとする．

役割指向の記述は，役割指向テンプレートジェネレータを用いて行う．また，システムの実装には，SOARSを参考にしたエージェント実行系を用いる．役割指向テンプレートジェネレータの記法により，役割指向をより明確に記述することが可能になる．SOARSを参考にしたエージェント実行系を用いることにより，役割指向により記述したシステムのエージェントやスポットを実行することができる．

役割指向テンプレートジェネレータとエージェント実行系によるシステムの記述と実装方法を示す．システムの実装は，汎用言語を用いて実装したエージェント実装系が必要である．汎用言語によるエージェント実行系と，役割指向テンプレートジェネレータによるシステムの役割指向記述を分離する（**図1**）．役割指向によるシステムの記述と，エージェント実行系を切り離すことにより，エージェント実行系を他のシステムの実装に利用することができる．（エージェント実行系を使用してシステムを実装するときのソースコードの構造をステージとして記述する必要がある．）また，役割指向テンプレートジェネレータによる役割指向記述を再利用し，システムの実装に使用するライブラリやフレームワークを変更することも可能である．

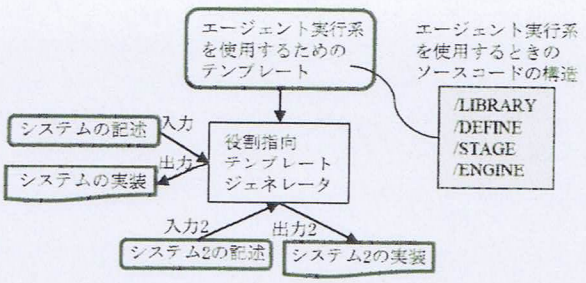

図1　役割指向テンプレートジェネレータによる実装

4.役割指向による並列分散システムの記述例

本節では，役割指向による並列分散システムの記述例について，並列分散システムの基本的な動作を行う，単純なシステムの実装を例として説明する．エージェント実行系の実装や，並列分散システムの実装のための記述が含まれるが，これらの記述言語として汎用言語であるClojureを用いている．実際には実装に用いる記述言語は，汎用言語であればこの言語のみに限らない[3].

例として，複数掛算同時実行を行う並列分散システムを考える．複数掛算同時実行を行う並列分散システムでは，クライアントが式Z=A*B+C*D+… を解くために，項の掛算をサーバに依頼する．サーバには掛算機能が実装してあり，サーバはクライアントから掛算の依頼を受けると掛算結果をクライアントに返す．クライアントにおいて，一つの項の掛算依頼と掛算結果の取得と他の項の掛算依頼と掛算結果の取得は，独立しているため並列に処理する．以下に示す記述例は，式Z=A*B+C*Dを解く場合であり，エージェントは2体である．

役割指向による複数掛算同時実行システムのクライアントの記述を示す．クライアントの全体の処理は，ステージという順序制約を表す概念により分割する．クライアントの処理は，/CONNECT（サーバとの接続生成），/SEND（掛算依頼），/RECEIVE（掛算結果取得），/SYNC（同期），/CALCULATE（項の和を求める），/TERMINATE（サーバとの接続終了）という順序の各ステージに分割する（リスト1）．1つのサーバへの処理（ネットワーク処理の/CONNECTと/TERMINATEや，並

列性のある仕事の依頼の処理の/SENDと/RECEIVE）を1体のエージェントが実行する．複数のエージェントとやりとりするオブジェクトをスポット（SOARSの概念）とし，クライアントを1つのスポットとする（図2）．

```
@@
:    ScenarioStage ;; ← Clientのステージ
/CONNECT
+
/SEND
+
/RECEIVE
+
/SYNC
+
/CALCULATE
+
/TERMINATE
+
```

リスト1　複数掛算同時実行システムのクライアントのステージの記述

エージェントやスポットは属性を持つことが出来る（リスト2）．エージェントは，サーバへの接続（属性socket）や並列処理のスレッド（属性future）を属性に保持することを示している．また，接続するサーバの情報（属性server-info）やサーバへ依頼する掛算のオペランド（属性operands），サーバから取得した掛算結果を格納する場所（属性place）を属性として保持する．エージェントの役割はWorkerRoleを設定する（属性roleにて設定する）．エージェントが存在するスポットは"client"であることが属性spotからわかる．

```
@agent-1
:    Agent
{
    (add-attr! self :server-info server-1)
    (add-attr! self :place :place-1)
    (add-attr! self :operands operands-1)
    (add-attr! self :socket nil)
    (add-attr! self :future nil)
    (set-attr! self :spot
                    (soars-spot "client"))
    (set-attr! self :role
                (soars-role "WorkerRole"))
}
```

リスト2サーバへの処理を実行するエージェントの記述

役割は継承することが可能である．ベースとなる役割のステージと関連付けられた処理を継承する．例えば，サーバへの接続を行う処理は，並列分散システムだけでなく，サーバ・クライアントシステムにおいて一般的に必要になる処理である．そこで，サーバとの接続を行うための処理を役割（ConnectorRole）としてモジュール化する．複数掛算同時実行システムのサーバへ仕事を依頼するエージェントの役割は，サーバへの接続の機能をもった役割ConnectorRoleを継承して作成する（役割WorkerRoleとする）．役割WorkerRoleにおいて/SENDと/RECEIVE，/SYNCのステージの処理を定義する．

役割WorkerRoleの記述をリスト3に示す．/RECEIVEは，各エージェントがサーバから依頼の結果を並列に取得する箇所である．/RECEIVEでは，Javaの Futureライブラリ（Futureは，マルチスレッドにおけるパターンの1つ）を利用して並列に処理する．そして，/SYNCにおいて/RECEIVEの処理の同期を行う．（/SYNCでは，/RECEIVEでの並列処理の結果を待つ場合がある．）/SYNCにおいて，クライアント（Spot）の属性にエージェントはサーバから取得した結果を格納する．

```
@WorkerRole
:    ConnectorRole
/SEND
{
    (send-request
      (ref-attr self :socket)
      (ref-attr self :operands))
}
/RECEIVE
{
    (set-attr! self :future
      (future (socket-recv
                (ref-attr self :socket))))
}
/SYNC
{
    (set-result! (ref-attr self :spot)
                 (ref-attr self :place)
                 (ref-attr self :future))
}
```

リスト3　サーバへ仕事を依頼するWorkerRole役割

複数掛算同時実行システムのクライアントにおける記述の一部を示した．本節にて示した記述以外には，役割ConnectorRoleやスポット(client)，スポットの役割が必要である．

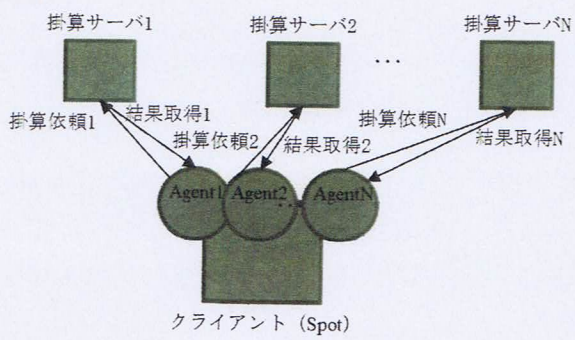

図2　エージェントとスポットによる複数掛算同時実行システムのクライアントの実装

5. 実験

複数掛算同時実行システムのクライアントの記述の一部は4節において示した．役割指向テンプレートジェネレータから，実装が生成されることを確かめる．

エージェント実行系を使用してシステムを実装すると

きのソースコードの構造をステージとして記述する必要がある．/LIBRARY，/DEFINE，/STAGE，/ENGINE の 4 つのステージである（リスト 4）．/LIBRARY ではシステムにおいて使う関数群を出力する．/DEFINE ではエージェントやスポット，役割の定義するコードを，/STAGE では役割にステージと関連付け処理が定義するコードを，/ENGINE では，エージェントやスポットを実行するためのコードを出力する．

```
@@
/LIBRARY
/DEFINE
/STAGE
/ENGINE
```

リスト 4　エージェント実行系を使うときのソースコードの構造を示したステージ記述

4 節において示した複数掛算同時実行システムのクライアントの記述とテンプレート（エージェント実行系を使用するためのテンプレート）から，以下の実装が生成される（リスト 5）．/DEFINE では，役割やスポット，エージェントを定義するためのコードが出力することが確認できる．/STAGE では，役割に処理（イタリック表記のコード：WorkerRole 記述の/SEND 内の行頭にタブ（空白）がある行のコード）を追加するコードが出力することがわかる．

```
...
DEFINE
(add-role! (make-role "WorkerRole"))
(add-role! (make-role "ClientRole"))
(define-spot! "client" code1)
(define-agent! "agent-1" code2)
...
STAGE
(update-stage! stage) ;; stage CONNECT
(add-action! (soars-role "WorkerRole") stage
                                        code3)
(update-stage! stage) ;; stage SEND
(add-action! (soars-role "WorkerRole") stage
    (send-request
      (ref-attr self :socket)
      (ref-attr self :operands))
)
...
```

リスト 5 役割指向テンプレートジェネレータから出力された複数掛算同時実行システムのクライアントの実装

6. 考察

本節では提案した役割指向による並列分散システムの記述や実装について考察する．5 節において，単純な並列分散システム（複数掛算同時実行システム）のクライアントの役割指向記述から，クライアント側のシステムの実装が生成できることを確かめた．役割指向を用いることによって，エージェントやステージ，役割に集中することができた．また，並列分散システムを構築する上で困難になりうる具体的なネットワーク処理や並列処理などの実装を隠蔽してシステムを設計することができた．これは，具体的な実装を役割指向により隠蔽することで，システムの実装に用いているライブラリやフレームワーク，エンジン（実行系）などの要素を，取り替えることが可能であることを示している．また，実装を隠蔽する役割指向記述において，システムをカスタマイズできる可能性がある．

並列分散システムの実装において,役割指向の利点を得るために，考慮しなければならないことがある．まず，システムにおいてエージェントをどのように捉えるのか，である．本研究では，エージェントを 1 つのサーバとやりとりをするオブジェクトとして捉えたが，サーバへ依頼する仕事を表すオブジェクトといった別の捉え方も可能である．また，ステージの構成方法もいくつかの可能性がある．本研究で実装した手法では，並列性やカスタマイズ性について考慮した．一方で，ステージの構成によっては，処理を必要以上に細かく分割してしまう可能性がある．また，ステージという概念がシステム構築に適さないことも考えられる．

さらに，役割指向という概念をシステムの実装と切り離して記述する方法についても考慮する必要がある．本研究では，システムを記述・実装するために，(1)役割指向という概念を使い，(2)エージェント実行系をシステムの実装に利用し，(3)役割指向テンプレートジェネレータをメタ記述言語のように使用した．役割指向の利点を得るために重要であった点は特に(1)，(3)が重要であり，これは，どのようにシステムの実装を隠蔽するかという観点である．役割指向の記述とシステムの実装を切り離すことは，困難でもある．本研究において使用した役割指向テンプレートジェネレータでは，エージェント実行系を使うときのパターンを定義した．そして，個々のシステムの実装において必要になる記述を役割指向記述と離して行った．このような役割指向記述と実装を分離するための仕組みが役割指向の利点を得るためには必要であると考える．

7. まとめ

本研究では役割指向による並列分散システムの有効な記述が可能であることを示した．役割指向による並列分散システムの記述方法と記述例を示し，役割指向の利点について考察した．役割指向の有効性を探るためには，応用事例を増やす必要がある．例えば，役割指向による記述のカスタマイズ性など，役割指向の利点に注目した応用事例である．そのためにも，役割指向を記述するための手法やツールの開発，改良が求められる．

文　献

[1] H. Deguchi, H. Tanuma, and T. Shimizu, “SOARS: Spot Oriented Agent Role Simulator - Design and Agent Based Dynamical System -,” in Proc. AESCS04, pp.49-56, 2004.

[2] 田沼英樹，“役割指向テンプレートジェネレータによる従来言語ルール記述と役割指向 ABM の結合,” JAWS2011.

[3] 柏木孝仁，佐々木晃，田沼英樹，“Scheme 言語をルール記述言語とした役割指向記述の試み,” 情報処理学会第 74 回全国大会，2012.