役割指向テンプレートジェネレータを用いたゲーミング・シミュレーション開発手法の提案

○女部田 雅俊　佐々木晃（法政大学）市川学（東京工業大学）

A Proposition of Development Method of Gaming-Simulation Using Role-Oriented Template Generator

* M. Onabeta , A. Sasaki (Hosei University) and M.Ichikawa（Tokyo Institute of Technology)

Abstract－ The role-oriented template generator gives a framework for realizing the agent base model without depending on implementation languages. The user can obtain ABM implementation effectively by using the template generator by composing the scenario-description that describes the individual simulation and base-description that describes a simulation engine. In this paper, we propose a development method of experiment environment for gaming-simulation by the programming technique based on the role by using a template generator. Our target is to develop a gaming-simulation effectively by adding the essential functions for gaming-simulation into base-description.

Key Words: Gaming-simulation, Role-Orientation, Agent based Modeling

1　はじめに

ゲーミング・シミュレーションは，人間がシミュレーションに介入し，ある時間の区切りごとに与えられた役割に応じた意思決定を行うことで，そのモデルで起きうる現象を疑似体験することが出来る手法であり，その性質から特に教育分野で有用なシミュレーションの一種である．ゲーミングは通常，社会，コミュニティ等における特定の側面をモデル化し，仮想的に構築した環境とその変化をシミュレートできるものである．参加者(プレイヤー)は，与えられた役割に応じた意思決定を行ったり，ゲームが規定するルールに従うことでゲーミングを進行させる．ゲーミングにおける意思決定を通じ，プレイヤーはゲーミングのモデルを疑似体験でき，またプレイヤー同士や環境との相互作用や，プレイヤーの行動の心理的変化などの観察を通じ，システムに対する深い洞察を引き出すことが可能である．現在では，ゲーミングはコンピュータを用いて行われることもあり，特にネットワークに複数台のコンピュータを接続し，ゲーミングを行うような場合，シミュレーションをサーバーで行い，意思決定を行う人間がクライアントとして参加するという形態をとることが多い．本手法では，特にこのネットワークを介したゲーミングを扱う．このような，ゲーミング・シミュレーションを構築するためには，サーバーやクライアントの準備や，サーバー・クライアント間で行われるメッセージの送受信の制御などの煩雑な処理が必要となってしまい，開発者の負担となってしまう．

本研究では，役割指向に基づくプログラミング手法によってゲーミングの実験環境を構築する手法を提案する．役割指向は，社会シミュレーション言語SOARS[1)2)]のベースとなる記述法で，ルールの実行順序を規定する「ステージ」，複数のルールで構成される「役割」および「エージェント」の3つを構成要素とする概念である．

本手法で用いる役割指向テンプレートジェネレータは，実装言語によらないエージェントベースモデル(ABM)の実現の枠組みを与える[3)]．テンプレートジェネレータは，シミュレーションエンジンを記述したベース記述と，個々のシミュレーションを記述したシナリオ記述を合成することで効果的にABMを得る[4)]．役割指向テンプレートジェネレータにおけるベース記述にゲーミングに必要な機能を加えることで，シナリオ記述に僅かな記述を追加するだけでゲーミングを構築することが可能となり，既存のモデルのゲーミング化や新規のゲーミングの開発を効果的に行うことを目標とする．

本稿では，提案手法の概要を説明した後，役割指向テンプレートジェネレータの枠組みを用いてゲーミング・シミュレーションを構築する手法を示す．そして，今回評価実験として実装したモデルを説明した後，本手法の考察を行う．

2　役割指向テンプレートジェネレータ概要

役割指向テンプレートジェネレータは，実装言語によらないエージェントベースモデル(ABM)の実現の枠組みを与える．テンプレートジェネレータは，シミュレーションエンジンを記述したベース記述と，個々のシミュレーションを記述したシナリオ記述を合成することで効果的にABMを得る．

役割指向テンプレートジェネレータを用いたエージェントベースシミュレーションの構築を，社会シミュレーションの実装を例に述べる．社会シミュレーションを汎用言語で実装する場合，個別のモデル構造を表す記述と，シミュレーションエンジン等の記述を分離することは容易ではなく，開発効率や分離できないことによる可読性の面に問題を残す．役割指向テンプレートジェネレータでは，シミュレーションオブジェクトのひな型やシミュレーションエンジンを記述したベース記述を用意し，これをすべてのシミュレーションの共通部分として用いる．個々のモデルは，シナリオ記述としてベース記述とは独立に記述し，役割指向テンプレートジェネレータがこれら二種の記述を合成しプログラムを得る．最終的に得るプログラムの記述言語はルール記述に用いた従来言語である(Fig. 1).

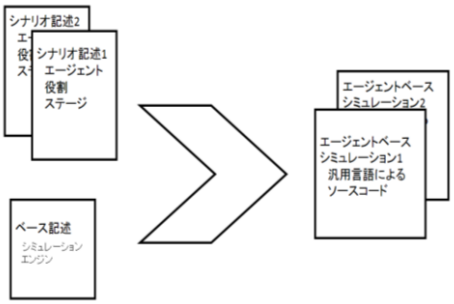

Fig. 1 Role-Oriented Template Generator

3 ゲーミング・シミュレーションの開発手法

提案手法は，役割指向の概念とベース記述に実装されたゲーミングとしての機能をシナリオ上で利用することで，開発者がサーバー・クライアント間で行われる煩雑な処理を意識することなくゲーミングの構築を可能とする．

本手法は，社会シミュレーション言語 SOARS で用いられている手法と同じ方法であるが，3 節で述べたように，役割指向テンプレートジェネレータの枠組みを用いることでシミュレーションを汎用言語で記述できるため, SOARS のようなドメイン特化型言語では手軽に行えないような処理も記述することが可能となる．

まず，役割指向の概念を用いることで，既存のモデルに対して容易にゲーミングの機能を拡張することが出来る．役割指向では，ステージの追加や役割の継承を柔軟に行うことが出来，シミュレーションの拡張・結合をよりモジュラーな記述で実現できる. そのため，役割指向で記述したゲーミングの要素を持たないシミュレーションモデルに対し，ゲーミングの環境を用意するためのステージや，人間がシミュレーションに介入するための役割を持ったエージェントなどを拡張することで，既存のモデルのゲーミング化を容易に行うことが出来る．

また，ベース記述にサーバーとクライアント間の通信などのゲーミングを行う上で必要な機能を拡張することで，シナリオ記述において必要最低限の記述でゲーミングを構築することが出来る．

さらに，サーバーやクライアントが行うべき処理を別に記述し，役割指向テンプレートジェネレータを用いてシミュレーションを作成する際に結合する．この様にすることで，サーバーやクライアントを作成するような共通部分と，モデル固有の処理を分離することが可能となり，シミュレーションの柔軟性や拡張性を高めることが出来る. 本手法の概要図を Fig. 2 に示す.

以下では，エージェントが役割に応じて決められた場所へと移動するシンプルなモデル（スポット移動モデル）を例としてゲーミングの構築手法を述べる．役割指向で記述されたこのモデルでは, 次の記述により，初期化ステージ(init)でスポットやエージェントの配置などを行った後，意思決定，移動(decision, move)の振る舞いを繰り返す．記述法の詳細は文献[4]を参照されたい.

```
@@
:       InitStage
/init
+
@@
:       MainStage
/decision
+
/move
+
```

ゲーミングのプレイヤーは，意思決定の段階でシミュレーションに介入し，エージェントの移動先を決定する．ここでは，Student，Worker，Person の三種類のエージェントがいると仮定する．プレイヤーは Person を通じてシミュレーションに介入する．以下に，役割指向テンプレートジェネレータを用いたスポット移動モデルの実装を示す.

```
@MoveRole
:       AgentRole
/move
{
        self.spot = self.next;
}
@StudentRole
:       MoveRole
/decision
{
  self.spot == Home ? self.next = School
                    : self.next = Home;
}
@WorkerRole
:       MoveRole
/decision
{
  self.spot == Home ? self.next = Office
                    : self.next = Home;
}
@Student
:       Agent
{
        self.spot = Home;
        self.role = StudentRole;
}
@Worker
:       Agent
{
        self.spot = Home;
        self.role = WorkerRole;
}
```

/move や/decision はステージを表し，そのステージを実行する際には括弧内の処理が行われる．また，MoveRole や StudentRole は役割を表しており，エージェントである Student や Worker はそれらの役割に応じた意思決定を行う.

以上をシナリオ記述とし，シミュレーションエンジンを記述したベース記述と合成することで，ゲーミングの要素を持たないスポット移動シミュレーションを得ることが出来る.

このスポット移動モデルを拡張し，ゲーミングを構築する際には，サーバーとクライアント間の通信が必要となる．本手法では，ゲーミングをより柔軟に行うため，シミュレーションを監視するためのエージェン

トであるオブザーバを作成する．ゲーミングを行う上で必要な処理をロールとしてまとめ，それをオブザーバに持たせることで，ゲーミングの拡張性やシナリオの可読性などを向上させる．

まず，シミュレーションを動作させるサーバーを構築する記述が必要となるシナリオ記述において，以下の GamingRole というオブザーバの役割を定義する．オブザーバにこの役割を持たせ，，prepare ステージを実行することでシミュレーションサーバーを用意することが出来る．必要な通信のための機能をベース記述に拡張することで，シナリオ記述での記述量を抑える．

```
@GamingRole
:       AgentRole
/prepare
{
    Gaming.serverCreate(1, clientList);
}
```

ここで，serverCreate()はベース記述に記述されており，サーバーの構築とクライアントの受け入れ準備を行うメソッドである．サーバーを作成する際に，参加させるクライアントの人数と，クライアントの情報を渡すことで，サーバー・クライアント間でのやり取りをより円滑に行う．

このスポット移動モデルにおいて，プレイヤーがシミュレーションに介入するタイミングは，意思決定(decision)ステージである．ベース記述に拡張されたゲーミングとしての機能を利用し，外部から意思決定を受け取る端の役割記述は，以下のように記述できる．

```
@PersonRole
:       MoveRole
/decision
{
  Gaming.sendMessage(self, ”next”);
}
```

この記述において，sendMessage()はベース記述に記述されており，接続されているクライアントのすべてにメッセージを送る．メッセージを受け取ったクライアントは意思決定を促すメッセージを表示し，プレイヤーの意思決定を待つ．すべてのクライアントのメッセージがそろった時点で次のステージへと移行する．本手法では，全てのクライアントからの返事を待つため，クライアントに意思決定を促すステージの直後にクライアントの返答を待つステージを作成する．この処理をベース記述に拡張することで，シナリオ記述では以下のようにして返答を待つ処理を利用できる．

```
@GamingRole
:       AgentRole
/wait
{
    Gaming.waitClientAnswer();
}
```

今回の実装では，クライアントの変数への値の代入はリフレクションを用いて行う．sendMessage に意思決定を行うクライアントと，値を代入するための変数名を渡すことで，指定した変数にプレイヤーが選択した意思を表す値を格納することが出来る．

このように，ゲーミングを行うためのオブザーバを作成し，プレイヤーがシミュレーションに介入するタイミングを定義した役割をエージェント Person に持たせることで，スポット移動モデルに人間が介入することが可能となる．

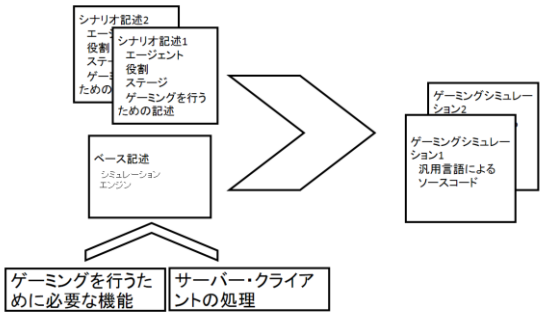

Fig. 2 Outline of Proposed Method

4 実装例

本手法の有意性を検証するため，囚人のジレンマとビールゲームの二つのゲーミング・シミュレーションの実装を行った．今回の実装は Java で行った．

4.1 囚人のジレンマ

囚人のジレンマはゲーム理論のモデルの一つであり，各プレイヤーが利得の大きい選択肢を選ぶ場合，協力した場合よりも悪い結果を招いてしまうゲームである．

本研究では，まず社会シミュレーション言語 SOARS に基づくゲーミングの要素を持たない囚人のジレンマモデルを構築した後，それをゲーミングへと拡張する．

今回作成したモデルでは，2 人の囚人(それぞれ A,B とする)ともに，半々の確率で自白か黙秘かを選択する．シミュレーションは，初期化ステージの後，意思決定ステージ，移動ステージを繰り返し実行する．次に，作成したモデルをゲーミングへと拡張する．今回のモデルでは，プレイヤーは囚人 B としてシミュレーションへと介入し，意思決定を行う．

まず，囚人のジレンマモデルをゲーミングへと拡張するため，オブザーバを定義する．今回オブザーバが持つ役割は，シミュレーションサーバーの準備とクライアントの返答の同期であり，3 節で述べた prepare ステージと wait ステージを持つ．prepare ステージを初期化ステージの後に，wait ステージを decision ステージの後に追加することで，サーバーの準備とクライアントの返答の同期を行うことが出来る．

次に，新たに PersonRole を定義し，エージェント Person に持たせることで，シミュレーションにプレイヤーを介入させる準備を行う．今回のモデルでプレイヤーがシミュレーションに介入するタイミングは，自白か黙秘かを選択する意思決定ステージである．3 節で述べたとおり，プレイヤーから意思決定を受け取るため，役割 PersonRole を以下のように記述する．

```
@PersonRole
:       LoggerRole
/decision
{
  Gaming.sendMessage(self, “decision”);
}
```

この記述により，囚人 B であるクライアントへと意

思決定を促すためのメッセージを送信し，メッセージを受けとったクライアントは，意思決定のためのウィンドウを表示する．プレイヤーが意思決定を行うと，選択された選択肢はサーバーに送信され，サーバー上でクライアントの意思として登録し，シミュレーションを進める．

実行結果を Fig. 3 に示す．プレイヤーは右下のダイアログを用いて意思決定を行い，サーバーで実行されているシミュレーションは，プレイヤーの決定をもとにシミュレーションを進行させる．

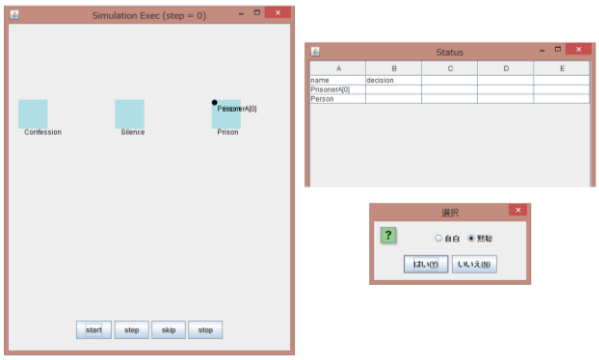

Fig. 3 Execution Result of Prisoner's Dilemma

4.2 ビールゲーム

ビールゲームはシステムダイナミクスの基本原則やシステム指向の重要性を学習できるモデルである．1 チーム 4 人のプレイヤーがそれぞれ小売，二次卸，一次卸，工場の 4 つのユニットを受け持ち，受注残や在庫コストを最小限にすることを目指す．シミュレーションは，それぞれのプレイヤーが受領(receive)，受注(accepting order)，発送(send)，発注(order)を繰り返し行うことで進行する．

本手法では，囚人のジレンマモデルと同じベース記述とオブザーバを用いてビールゲームを作成する．1 チーム複数人で別の端末から意志決定を行うことを想定し，複数のクライアントからの返答を同期しつつシミュレーションを進める．3 節で述べたように，本手法ではオブザーバが持つ wait ステージを用いることでこの問題を解決する．

今回のモデルでプレイヤーがシミュレーションに介入するタイミングは，上流への発注数を決定する発注の部分である．複数のクライアントで行われるようなゲーミングでは，意思決定の際に全てのクライアントからの返答を待たなければシミュレーションを進めることが出来ないため，order ステージを実行した後，wait ステージを実行する．wait ステージでは，クライアントが返答を行ったかどうかを監視し続け，全てのクライアントからの返答がそろった時点で監視を終了し，シミュレーションを次のステージへと進める．

5 考察

提案手法を用いることで，役割指向で記述された既存のモデルのゲーミング化を効率的に行うことが可能であることが分かった．3 節で述べた通り，役割指向の概念を用いることでステージの追加などを柔軟に行うことが可能であり，既存のモデルにゲーミングを行う準備のためのステージを追加や削除を行うことで，既存のモデルのゲーミング化や，役割指向で記述されたゲーミングを，ゲーミングの要素を持たないシミュレーションとして動作させることが可能である．

役割指向テンプレートジェネレータの枠組みを応用し，ゲーミングに必要なサーバーやクライアントの作成，メッセージの送受信などの制御をすべてのモデルの共通部分としてベース記述に拡張することで，シナリオ記述における記述量を必要最低限に抑えることが可能となる．そのため，ゲーミングに必要な機能を拡張したベース記述を用いて開発を行うことで，新規にゲーミングを開発する際でも効率的に構築を行うことが出来ると考えられる．

今回の実装は Java を用いて行ったが，2 節で述べたように，役割指向テンプレートジェネレータはルール記述に用いた従来言語で記述されたプログラムを生成する．そのため，3 節で述べたように，今回意思決定を代入するための変数を柔軟に変更できるようにするために，リフレクションを用いた．ように，必要とするライブラリや言語仕様に合わせて実装言語を変えることで，より効率的なゲーミング・シミュレーション開発を行うこともできると考えられる．

6 おわりに

本研究では，役割指向テンプレートジェネレータを利用し，役割に基づくプログラミング手法によってゲーミングの実験環境を構築する手法を示した．本手法では，ゲーミングを行う際に必要となる，サーバーやクライアントの作成やサーバー・クライアント間で行われるメッセージの送受信の制御をモデルの共通部分としてベース記述に拡張した．この手法により，シナリオ記述におけるゲーミングの記述を必要最低限に抑えることが可能となり，既存のモデルをゲーミングへと拡張することや，新規にゲーミングを開発することをより効果的に行うことが出来る．

今後の課題は，サーバーやクライアントが行うモデル固有の処理のより効果的な拡張法や，ゲーミング・シミュレーションのより本格的な実装に関する研究などが挙げられる．

謝辞

本研究の一部は，科研費(課題番号 23700043, 23500034)の助成を受けた．

参考文献

1) 田沼英樹，出口弘，エージェントベース社会シミュレーション言語 SOARS の開発，信学会論文誌，Vol.J90-D, No.9, 2415/2422 (2007)

2) Manabu Ichikawa, Hideki Tanuma, Yuhsuke Koyama and Hiroshi Deguchi, SOARS for simulations of social interactions and gaming, Introduction as a social microscope, *Proc. 38th Annual Conference of the International Simulation and Gaming Association*, P-36, (2008)

3) 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，: 合同エージェントワークショップ＆シンポジウム 2011(JAWS2011).

4) 佐々木晃，柏木孝仁，田沼英樹，役割指向テンプレートジェネレータを利用したエージェントシステムの効果的な設計と実装: 合同エージェントワークショップ＆シンポジウム 2012(JAWS2012)