MAGCruise: マルチエージェントモデルに基づくゲーミング環境

中島　悠[†a)]　菱山　玲子[††]　中口　孝雄[†††]

MAGCruise: Gaming Environment Based on Multiagent Model

Yuu NAKAJIMA[†a)], Reiko HISHIYAMA[††], and Takao NAKAGUCHI[†††]

あらまし　社会に生じる様々な問題を分析し理解するため，社会的なコミュニケーションを対象としてマルチエージェントモデルに基づくゲーミングが行われている．人間と人間，人間と環境の関係性にまつわる社会的な問題のモデル化には，マルチエージェントモデルが適している．本研究の目的は，シミュレーションが対象とする領域のドメイン知識を有する専門家が，自ら問題をゲームとして記述し，ゲーミングを実施できるようにすることである．この問題領域の専門家はソフトウェア技術者とは限らない．本研究では，ゲームのプロセスを人間またはコンピュータに操作されるプレーヤ群が実行するワークフローとして捉え，これを簡易に定義できるゲームシナリオを設計した．また，そのシナリオと実験設定を Web システムに与えるだけでオンライン参加型のゲーミングが実施できる環境を構築した．被験者実験により，ソフトウェア技術者ではない問題領域の専門家が，本環境を用いて順次手番や同時手番を組み合わせたゲームのシナリオを記述し，ゲーミングを実施できる可能性を示した．

キーワード　マルチエージェントシミュレーション，ゲーミング，参加型シミュレーション，Web ベースラーニング

1. まえがき

マルチエージェントシミュレーション (MABS: Multiagent-based simulation) では，ソフトウェアとして記述されたエージェントが活動することでシミュレーションが実行される．交通システムや経済・市場モデル，意思決定メカニズムを扱う分野において，MABS を用いて社会的なインタラクションを分析することで，社会の問題を分析し理解することが試みられている．このような問題の分析には，個々の人間をエージェントとしてモデル化してその間のインタラクションを計算するマルチエージェントモデルが適している．

MABS については近年，頻繁に解説記事 [1], [2] などでも紹介され，盛んに研究が進められている．MABS の分野では，仮想空間においてエージェントだけではなく人がシミュレーションに参加する参加型シミュレーション [3], [4] や，ロールプレイを利用して参加者の訓練や学習を促すゲーミングシミュレーションの研究が進められている．

ゲーミングでは，はじめに，実験実施者となる研究者や実践者が対象とする問題をゲームとして定式化する．実験実施者は，ゲームを実施することで問題を再現し，参加者の振る舞いを観察する．参加者は，問題における何らかの役割を割り当てられ，問題に巻き込まれる体験をする．ゲーミングを実施することで，実験実施者は問題の理解の深化や解決のための方略の探索をし，参加者は自身の体験を通じた問題の捉え直しをすることができる．

これまでゲーミング環境は，主に教育ツールないし研究実験ツールとして開発・活用され，発展してきた．ゲーミングシミュレーションを用いる研究の多くは，それぞれの問題にあわせて個別のゲーミング環境を新規に開発することが多く，その実施は高コストであった．それに対して，ゲーミング環境の開発及びゲーミ

† 東邦大学理学部情報科学科，習志野市
Department of Information Science, Toho University, Narashino-shi, 274–8510 Japan
†† 早稲田大学理工学術院，東京都
Faculty of Science and Engineering, Waseda University, Tokyo, 169–8555 Japan
††† 京都大学大学院情報学研究科，京都市
Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan
a) E-mail: yuu.nakajima@is.sci.toho-u.ac.jp
DOI:10.14923/transinfj.2014SWP0013

ングの実施コストを縮小するため，記述できるゲームを経営ゲームなど特定領域の問題のみに限定することでゲームの容易な生成を可能としている環境もある [5], [6].

本研究の目的は，多様な問題群を扱う研究者が，ゲームを記述し，ゲーミングを実施するコストを小さくすることである．それを実現するため，本研究では以下の二つの課題に取り組んだ．

- **ゲーム定義の設計**

マルチエージェントモデルに基づくゲーミングの対象となる問題は，個人と他者や個人と環境との関係性にまつわる社会的なものが主であり，複数の人間が協調して課題を遂行するプロセスを形成する．このようなプロセスを簡易に記述できる仕組みが必要である．

- **オンライン参加型のゲーミングシステムの構築**

ゲームで対象とする問題のドメイン知識を有する専門家はソフトウェア技術者ではないことが多い．そこで，記述したゲーム定義を使ってオンライン参加型のゲーミングを簡易に実施できるシステムが必要である．

2. ゲーミング

ゲーミングないしゲームとは何かということについては，様々な見方がある．システムを解釈するためのシミュレーションツール，ゲーム理論の拡張としての記述方法，ロールプレイによる社会的プロセスとの相関を把握する方法，学際融合を実現する手法，大衆的な娯楽，教育的な手法など多様な説明が存在する．1974 年にゲーミングに関する体系的な解説 [7] を著した Richard D. Duke は，ゲーミングを「コミュニケーションの一様式」と結論づけている．

ゲーミングでは，実験実施者は，人々の行動やその要因を明らかにするため，問題の本質的な部分をゲームとして切り出し，参加者に提示する．参加者は，問題における何らかの役割が割り当てられ，問題を体験する．この体験は，問題の現場における体験と同一ではなくても，より近い体験であることが期待される．

ゲーミングは，参加者から見ると，社会や組織の問題を割り当てられた役割に基づいて体験するものと言える．また，実験実施者から見ると，ゲームにおける参加者の行動の観察やフィードバックの分析を通じて，問題の理解を深めるものと言える．

ゲーミングは人が操作するプレーヤのみにより実行される場合もあれば，人に加えて実験実施者により記述されたプログラムが操作するプレーヤを含む系として実行される場合もある．この点で，ゲーミングは，ロールプレイを利用した MABS の一形態であると言える．

ゲーミングは，問題のモデル化の適切性や結果の妥当性の評価に困難を抱えている．これらの課題は，MABS が抱える課題とも共通しており，MABS と両輪での研究が必要である．

また，ゲーミングの運用には，社会への深い洞察力や人や組織の役割・環境を記述する力が要求される．この点でも，被験者による実験を積極的に採用している心理学や実験経済学，フィールドの問題を扱う社会学分野の専門家との連携が不可欠であると言える．

3. ゲーム定義

本研究で対象とする問題は，心理学や経済学，交通工学の分野で MABS や被験者実験が実施されているような問題である．つまり，社会や集団をマルチエージェントの系として捉え，そこにおけるエージェントのインタラクションやエージェントの意志決定を扱う諸問題をゲームとして扱うことを想定している．

本研究では，ゲームのプロセスを人間またはコンピュータに操作されるプレーヤ群が実行するワークフローとして捉えて，ゲームを定義する．ゲームの定義は主に，1) ゲームのプロセスのワークフローを記述するゲームシナリオ，2) プレーヤの行動や環境の変化を具体的に計算するプレーヤ内部モデル及び環境モデルの二つから構成される．

3.1 ゲームシナリオ

ゲームシナリオは，プレーヤが協調して課題を遂行するプロセスをワークフローとして記述するものである．言い換えると，ゲームシナリオには，プレーヤが取るべき行動とその実行順序が記述される．図 1 は，ゲームシナリオによるプレーヤの制御を示すものである．ゲームシナリオの実行エンジンは，ワークフロー

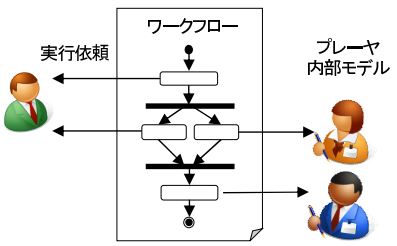

図 1 ゲームシナリオ
Fig. 1 Game scenario.

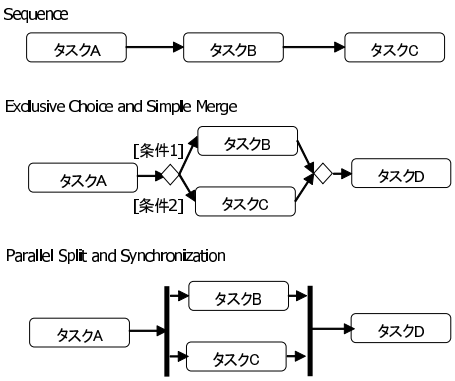

図 2　ワークフローの Basic Control Flow Patterns
Fig. 2　Basic control flow patterns of workflow.

を解釈し，プレーヤに対して行動の実行依頼をする．行動の実行依頼の処理はプレーヤ内部モデルが受けもつ．

ビジネスプロセスにおいて，ワークフローは，UML (Unified Modeling Language) 2.0 アクティビティ図や BPMN (Business Process Modeling Notation) により図示されることが多い．

Aalst らは，ワークフローの Basic Control Flow Patterns として，Sequence, Exclusive Choice, Simple Merge, Parallel Split, Synchronization をあげている [8]．Exclusive Choice と Simple Merge，Parallel Split と Synchronization は，それぞれ組み合わせて用いられることが多い．これらの例をアクティビティ図で図示したものが図 2 である．

社会や集団における人の振る舞いを扱うゲームのプロセスは，同時手番ゲームである戦略型ゲーム，順次手番ゲームである展開型ゲームに大別される．このようなゲームのプロセスは，同じ人の営みであるビジネスのプロセスと同様にワークフローとして記述しやすいと考えられる．ゲームシナリオでは，タスク，ステージ，ラウンドという要素を使って，ゲームをワークフローとして記述する．

- タスク

タスクは，ある一つの行動を表すものである．タスクにはプレーヤが実行するものと環境が実行するものがある．タスクはアクティビティ図におけるアクションに相当する．

- ステージ

ステージはタスクの実行制御をするため，幾つかのタスクまたはステージを束ねたものである．ステージは，アクティビティ図において幾つかのアクションに制御構造を与えてまとめたものだと言える．

ステージには Basic Control Flow Patterns と対応して，順次実行ステージ，排他実行ステージ，並列実行ステージの三種類が存在する．

順次実行ステージは，順序付けされたタスクの組を表すもので Sequence パターンに相当する．このステージが開始されると最初のタスクが実行される．前のタスクが完了すると次のタスクが実行される．全てのタスクが終了するまでこれが繰り返される．

排他実行ステージは，排他的に実行されるタスクの組を表すもので，Exclusive Choice パターンと Simple Merge パターンに相当する．このステージが開始されると，プロセスのパスはタスクごとに複数に分割され，条件に応じたタスクが排他的に実行される．タスクの実行が完了すると，分割されたパスが単一のパスに結合される．

並列実行ステージは，同時に実行され終了後に同期がとられるタスクの組を表すもので，Parallel Split パターンと Synchronization パターンに相当する．このステージが開始されると，プロセスが通るパスは複数のタスクが同時に実行されるように複数のパスへと分割される．そのステージが含む全てのタスクが終了すると，並列パスの同期が行われ分割されたパスが結合される．

シミュレーションとゲーミングの大きな違いは，何人かのプレーヤが人間により操作されていることである．一般に人間の意志決定はコンピュータによる意志決定より時間がかかる．そこで，プレーヤが同時に意志決定可能な箇所は積極的に並列化する必要が出てくる．例えば，多数決のような同時手番ゲームを考える．もし，全プレーヤがコンピュータにより操作されるプレーヤであった場合は，このプロセスを順次実行のプロセスとして実行しても問題は起こらないだろう．しかし，全プレーヤが人間により操作されるプレーヤであるような場合は，並列実行のプロセスとして実行しなければ長い時間が必要になる．

- ラウンド

ゲームシナリオはラウンドの集合として定義される．ラウンドは記述順に実行される．ラウンドは幾つかのステージを束ねたものであり，ラウンド内のステージは記述順に実行される．ステージがタスクの実行制御をするために存在するのに対して，ラウンドはステージを概念的にまとめるために存在する．

3.2 プレーヤ内部モデルと環境モデル

プレーヤ内部モデルと環境モデルは，ゲームシナリオから実行を依頼される行動 (タスク) を実装するものである．ゲームシナリオで用いられる行動 (タスク) は，ゲームのワークフローを構成する語彙であり，具体的な実装を伴うものではない．この語彙は，プレーヤ内部モデルと環境モデルとして実装される．

シナリオ記述言語 Q [9] では，問題領域の専門家と計算機の専門家が協調して作業ができるように，インタラクションをデザインする層とエージェントの内部を実装する層が明確に切り離されている．本環境において，タスクを処理する順序を示すゲームシナリオとタスクを処理するプレーヤ内部モデル，環境モデルを明確に分離しているのも同じ思想による．

プレーヤ内部モデルや環境モデルを実装する際には，プレーヤオブジェクトと環境オブジェクトを利用できる．プレーヤオブジェクトは，プレーヤごとの個別の情報を蓄えるものである．それに対して，環境オブジェクトはゲーム内で共有されるただ一つのオブジェクトである．プレーヤオブジェクトは，他のプレーヤに同期型のメッセージを配送できる．配送されたメッセージは対象プレーヤのメッセージボックスに蓄えられる．プレーヤ間のデータの受け渡しはこのメッセージの授受で表現される．

コンピュータが操作するプレーヤの内部モデルと異なり，人間の参加者が操作するプレーヤの内部モデルには，ゲームの文脈を人間が見ている画面に出力し，それに対する意志決定を人間に入力させる入出力機能が必要となる．それらを支援するため，参加者が操作する Web ブラウザに HTML フォームを表示し，その結果を内部モデルで取得するための関数群を用意した．

3.3 記 述 例

経済学の分野では，複数のプレーヤが順に意思決定をする順次手番ゲームは展開型ゲームと呼ばれ，MABS や被験者実験を用いた研究の対象となっている [10]．ここでは，経済学の分野でよく知られている展開型ゲームの一種である最後通牒ゲーム [11] を取りあげ，ゲーム定義の記述を説明する．

最後通牒ゲームの概要は「ゲーム実施者から P_1 にいくらかのお金が渡され，P_1 はこれを R_1 と 2 人で分ける．その際，分配する額は P_1 が決めることができるが，もし R_1 が受け取りを拒否したら，P_1 も R_1 も共に何も受け取ることはできない」というものである．

この最後通牒ゲームにおいて「受け取り手のプレーヤを R_1 と R_2 の 2 人に増やし，P_1 がそのそれぞれに対して最後通牒を伝え，2 人の応答が集まるとそれぞれとお金の分配を計算する」という拡張をしたものを考える (図 3)．

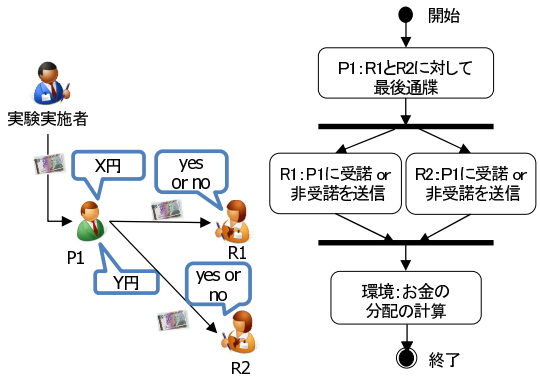

図 3 拡張された最後通牒ゲーム
Fig. 3 Extended ultimatum game.

本環境では，ゲーム定義のインタプリタは Scheme の処理系に実装されている．この拡張された最後通牒ゲームのシナリオは以下のように記述される．

```
(define (def:game-scenario)
  (def:round
    (def:stage 'ultimatum
      (def:task 'P1 'issue-ultimatum))
    (def:parallel-stage 'yes-or-no
      (def:task 'R1 'vote-yes-or-no)
      (def:task 'R2 'vote-yes-or-no))
    (def:stage 'divide
      (def:task 'divide-money))))
```

最後通牒ゲームは最後通牒ステージ (ultimatum)，応答ステージ (yes-or-no)，配分ステージ (divide) というステージからなり，順に実行される．

最後通牒ステージは，順次実行ステージである．def:stage 関数は，第 1 引数がステージ名，以降の可変長引数が順次実行されるタスクである．このステージでは，P_1 が最後通牒 (issue-ultimatum) を実行する．def:task 関数は，第 1 引数が行動をするプレーヤ (省略した場合は環境)，第 2 引数が行動に該当する関数の名前である．

応答ステージは，並列実行ステージである．つまり，R_1 と R_2 がそれぞれ並列して最後通牒に対して Yes か No で応答 (vote-yes-or-no) する．def:parallel-stage 関数は，第 1 引数がステージ名，以降の可変長引数が並列実行されるタスクである．

配分ステージでは，お金の分配を計算するタスク (divide-money) が環境により実行される．

上述のように，ゲームシナリオとして，ステージと

図 4　P_1 のプレイ画面
Fig. 4　Screen of P_1.

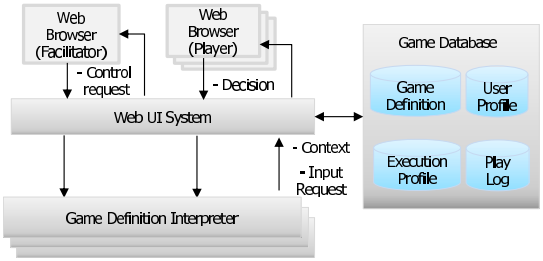

図 5　ゲーミングシステム MAGCruise の構成
Fig. 5　Architecture of gaming system MAGCruise.

タスクを組み合わせることでゲームのワークフローを記述する．各タスクにはプレーヤ内部モデルに属する関数 (issue-ultimatum, vote-yes-or-no) または環境に属する関数 (divide-money) が割り当てられており，各タスクはワークフローで指定された順に実行される．以下では，プレーヤ内部モデルに属する関数の例として，P_1 が実行する関数 issue-ultimatum を説明する．

```
(define (issue-ultimatum ctx self)
  (ui:request-input self:name
    (ui:form "R1 にいくらを分け与えますか？"
      (ui:val-input 'proposition))
    (lambda (input)
      (set! self:proposition input)
      (send-message 'R1 (cons 'proposition input)))
    ;;$R_2$に対しても同様の処理をする．)
```

ここで P_1 は,「R_1 にいくらを分け与えますか？」と参加者に入力を求めるフォームとその入力結果 (input) を引数として受けとるコールバック関数を登録している．ui:request-input 関数は，第 1 引数が入力を求めるプレーヤ名，第 2 引数が入力フォーム，第 3 引数がコールバック関数である．図 4 は，P_1 が issue-ultimatum 関数を実行したときの表示画面例である．

コールバック関数が呼び出されると，P_1 は自己の proposition 属性に入力された提案金額を格納する．プレーヤ内部モデルの実装では，環境オブジェクト (ctx) と自身を表すプレーヤオブジェクト (self) を利用できる．つづいて，P_1 はメッセージの proposition 属性にも提案金額を格納し，R_1 にメッセージを送信する．

4. ゲーミングシステム

MABS を開発し，実行するため数々のシミュレーション言語やツールキットが提供されてきた [2]．しかし，コンピュータにより操作されるエージェントだけが存在するシミュレーションと，そこに人間が参加するゲーミングとの差は大きい．本研究では，ゲーミングのための開発コスト及び実施コストを低減するため，前節までに述べたゲーム定義を用いてオンライン参加型のゲーミングを実施する環境 MAGCruise(注1) を開発した．

MAGCruise では，実験実施者は，ゲーム定義を記述し，それを Web システムにアップロードすることで，オンライン参加型のゲーミングを実施できる．これまでに本環境を用いて，匿名性が最後通牒ゲームに与える影響を観察するために 138 名 (69 組) が参加する実験を実施してきている [12]．また，他の事例に関する概要は [13] に紹介されている．

ゲーミングシステム MAGCruise の構成を図 5 に示す．MAGCruise では，ゲーム実施者 (facilitator) とゲーム参加者 (player) は Web システム (Web UI system) に Web ブラウザでアクセスする．ゲーム実施者はデータベース (game database) に保存されたゲーム定義やユーザ情報などを選択し，ゲームの開始を要求する．ゲーム定義のインタプリタ (game definition interpreter) は，その情報を用いてゲームセッションを作成し，参加者を募集する．ゲームを実行するのに必要な参加者が集まった時点で，ゲームが開始される．

ゲームにおいて参加者は，Web システムからゲームの文脈 (context) を受け取り，Web システムからの要求に応じて意志決定 (decision) を入力する．これが繰り返され，ゲーム終了の条件が満たされるまでゲームシナリオの実行が進んで行く．

ゲーム実施者は，図 6 に示すような画面で，ゲームの実施を管理できる．また，ゲーム実施者は，ゲームが終わった後，データベースに保存されたゲームの設定情報やゲームの結果を閲覧することができる．

MAGCruise では，ゲーム定義のインタプリタは Java で実装された Scheme の処理系である

(注1)：MAGCruise, http://www.magcruise.org/jp/

図 6　ゲーム実施者の管理画面
Fig. 6　Screen of facilitator.

Kawa(注2) を用いて実装されている．そのため，Java や Java で書かれたライブラリとの連携が容易である．

5. 議　　論

Repast(注3) や MASON(注4) など後続の多くのシミュレーション記述言語に影響を与えてきたシミュレーション記述言語 Swarm(注5) は，セルオートマトンモデルを基にしたものである．それに対して，本環境のように社会的なインタラクションを扱うシミュレーション記述言語としてシナリオ記述言語 Q [9] と社会シミュレーション言語 SOARS [14] を取りあげて議論する．

シナリオ記述言語 Q は，一体一体のエージェントに他者や環境と相互作用するためのインタラクションプロトコル割り当てることで，エージェントを制御するモデルをもつ言語である．Q の実行環境は，インタラクションの管理のみを行うものであり，他のシミュレータに接続して使用される．

Q のようにエージェント間のインタラクションプロトコルを定義することでエージェントを制御する方式は，不特定のイベントに駆動されるプレーヤの振るまい (例えば，不特定のタイミングで他のプレーヤから話しかけられ，それに対して返答をする) が主であるときに適している．本環境には，インタラクションプロトコルによりプレーヤを制御する枠組みも実装されている．

Q で記述されるエージェントシナリオはインタラクションプロトコルをエージェントごとに記述してエージェントを制御するものであるのに対して，本環境で記述するゲームシナリオはプレーヤ全体が従うワークフローを記述するものである．この点でこの二つのシナリオが対象を制御する視点は大きく異なると言える．

社会シミュレーション言語 SOARS は，社会や組織の理解とそのデザインを強く意識した社会シミュレーション言語である．SOARS では，エージェントの行為を社会的役割としてモデル化して記述する．SOARS は，社会シミュレーション言語としての機能だけでなく，モデル開発のためのビジュアルなシェルやゲーミングビルダーと言った機能をもつ [15]．ゲーミングビルダーは，プレーヤの属性を読み書きする Web ページを作成する機能と，その入出力をプレーヤ間で相互に反映させるサーバ機能をもつモジュールである．

SOARS は，社会的役割という概念を含むシミュレーションを独自のモデルを用いて記述する．SOARS の諸機能は，このモデルに基づいて設計され，提供されている．それに対し，本環境は，プレーヤが協調して課題を遂行するゲームというプロセスを一般的なワークフローのモデルを用いて記述する点が特徴的である．

6. 実　　験

6.1 設　　定

本環境におけるゲーム定義とゲーミング実施の容易性を確認するため，最後通牒ゲームの派生ゲームを課題として被験者実験を実施した．

はじめに，被験者はチュートリアルとして，基本的なワークフロー及び最後通牒ゲームのサンプルコード，ゲームシナリオとプレーヤ内部モデル・環境モデルのリファレンス，開発環境に関して 40 分間の説明を受けた．つづいて，説明された内容の動作確認や質疑をする練習時間が 10 分間与えられた．その後に，被験者はゲーム定義とゲーミングを実施する課題に取り組んだ．

課題として与えたゲームの概要とそれが含む実行制御の種類の一覧を表 1 に示す．課題としたゲームは，順次手番，同時手番，またはその両者を組み合わせたゲームである．課題となったゲームの実行制御は，順次実行だけのもの，順次実行と排他実行を組み合わせたもの，順次実行と並列実行を組み合わせたもの，順次実行と排他実行と並列実行を組み合わせたものの 4 種類であった．

被験者には，これらのゲームの説明文とアクティビ

(注2)：Kawa, http://www.gnu.org/software/kawa/
(注3)：Repast, http://repast.sourceforge.net/
(注4)：MASON, http://cs.gmu.edu/~eclab/projects/mason/
(注5)：Swarm, http://www.swarm.org/

表 1　課題としたゲーム
Table 1　Game list for task.

ゲーム	実行制御	概要
1	順次実行	プレーヤ A，B が参加する．A は B と最後通牒ゲームをした後で，A は B と再び最後通牒ゲームをする．
2	順次実行	プレーヤ A，B，C が参加する．A は B と最後通牒ゲームをした後で，A は C と最後通牒ゲームをする．
3	順次+排他実行	プレーヤ A，B が参加する．A の通牒に対して B が拒否をしたとき，A はもう一度だけ B と最後通牒ゲームができる．
4	順次+排他実行	プレーヤ A，B，C が参加する．A が B と最後通牒ゲームをする．B が受諾しなかった場合は，そこでゲームは終了する．B が受諾したとき，B は受けとった額を元に C と最後通牒ゲームをする．
5	順次+並列実行	プレーヤ A，B，C が参加する．A と B，A と C は独立に並列して最後通牒ゲームをする．
6	順次+並列実行	プレーヤ A，B，C が参加する．A と B，A と C は並列して，通牒及び応答をする．分配は B と C の応答が完了してから実行する．**3.3** の拡張された最後通牒ゲームと同じである．
7	順次+排他+並列実行	プレーヤ A，B，C が参加する．A と B，A と C は並列して，通牒及び応答をする．分配は B と C の応答が完了してから実行する．B が通牒を拒否した場合は，A はもう一度だけ B と最後通牒ゲームを行う．
8	順次+排他+並列実行	プレーヤ A，B，C が参加する．A と B，A と C は独立に並列して最後通牒ゲームをする．C が通牒を拒否したときのみ，A はもう一度だけ C と最後通牒ゲームを行う．

ティ図，チュートリアル資料，リファレンスが与えられた．被験者はタスクとステージを組み合わせてワークフローを表現するゲームシナリオを書くことが求められた．ゲームシナリオから呼び出されるプレーヤ内部モデルと環境モデルの実装は実験実施者より与えられた．

被験者は，それぞれの端末でゲームシナリオを記述し，そのシナリオを元にゲーミングを実行し，動作を確認した．被験者は 45 分の制限時間内に，できるだけ多くのモデルのゲームシナリオを正しく記述することが求められた．

6.2 結　　果

実験後のアンケートで，被験者にプログラミングのおよその学習時間を回答してもらった．その回答を元に，被験者をプログラミングに関して，未経験，経験少，経験多の 3 グループに分けた (表 2)．合計 12 名の被験者のうち，未経験 (学習時間が 0) の被験者が 3 名，経験少の被験者 (学習時間が 120 時間以下) が 3 名，経験多 (学習時間が 200 時間以上) の被験者が 6 名であった．

表 2　被験者のプログラミング経験
Table 2　Programming experience of human subjects.

プログラミング経験	学習時間	人数
未経験	0 時間 (3 名)	3
経験少	20 時間，40 時間，120 時間	3
経験多	200 時間 (5 名)，350 時間	6
合計		12

表 3　課題に含まれる実行制御の種類とプログラミング経験ごとの正答率と平均所要時間
Table 3　Results of tasks grouped by control flow type and programming experience.

実行制御	プログラミング経験	正答率 (=正答数/回答数)	平均所要時間 (分)
順次実行	未経験	1.00 (=6/6)	3
	経験少	1.00 (=6/6)	5
	経験多	1.00 (=12/12)	2
順次+排他実行	未経験	1.00 (=6/6)	5
	経験少	0.83 (=5/6)	3
	経験多	1.00 (=12/12)	4
順次+並列実行	未経験	0.83 (=5/6)	4
	経験少	0.83 (=5/6)	3
	経験多	1.00 (=12/12)	4
順次+排他 +並列実行	未経験	0.50 (=3/6)	5
	経験少	0.67 (=4/6)	7
	経験多	0.92 (=11/12)	7

課題に含まれる実行制御の種類とプログラミング経験ごとに，正答率 (正答数/回答数) と平均所要時間 (単位は分) をまとめたものを表 3 に示す．平均所要時間は，正答した場合の課題を解くのに要した時間の平均である．この実験では，被験者に課題を早く解くことを求めなかったため，平均所要時間は被験者のパフォーマンスを直接示すものではないが，参考のために付記する．

順次実行，順次実行と排他実行，順次実行と並列実行が課題に含まれる場合では，プログラミング経験にかかわらず，8 割以上の正答率が得られた．順次実行，排他実行，並列実行の三つの組合せが課題に含まれる場合では，プログラミング経験が多い被験者では 9 割以上の正答率があり，プログラミング経験がないまたは少ない被験者でも，2 分の 1 から 3 分の 2 の正答率があった．この結果は，ソフトウェア技術者ではない問題領域の専門家が，本環境を用いて順次手番や同時手番を組み合わせたゲームのシナリオを記述し，ゲー

ミングを実施できることを十分に示唆していると言える．

7. むすび

社会に生じる様々な問題を分析し理解するため，社会的なコミュニケーションを対象としてマルチエージェントモデルに基づくゲーミングが行われている．本環境は，このようなゲームの開発とゲーミングの実施を容易にするものである．

本研究では，ゲームのプロセスを人間またはコンピュータに操作されるプレーヤ群が実行するワークフローとして捉え，これを簡易に記述できるゲーム定義を設計した．また，ゲーム定義と実験設定を Web システムに与えるだけでオンライン参加型のゲーミングを実施できる環境を構築した．

今後は，ゲーミングの結果を使ってゲームのシナリオやプレーヤのモデルを洗練させていく手法を検討していきたい．

謝辞　本研究は科学技術振興機構 RISTEX「サービス指向集合知に基づく多言語コミュニケーション環境の実現」及び日本学術振興会科研費基盤 (S) (24220002, 平成 24 年度～28 年度) の助成を受けた．また，システム開発に関して協力頂いた Ageet Singapore Pte. Ltd. 樋口哲朗氏に感謝の意を表す．

文　献

[1] “特集 エージェント,” 人工知能学会誌，vol.28，no.3，pp.358–423, 2013.
[2] 市川　学，後藤祐介，“社会システムの研究動向 5—研究のためのツール–シミュレーション言語の特徴と比較,” 計測と制御，vol.52, no.7, pp.595–600, 2013.
[3] D. Torii, T. Ishida, and F. Bousquet, “Modeling agents and interactions in agricultural economics,” Proc. 5th International Joint Conf. on Autonomous Agents and Multiagent Systems, pp.81–88, 2006.
[4] Y. Murakami, T. Ishida, T. Kawasoe, and R. Hishiyama, “Scenario description for multi-agent simulation,” Proc. 2nd International Joint Conf. on Autonomous Agents and Multiagent Systems, pp.369–376, 2003.
[5] T. Terano, H. Suzuki, Y. Kuno, and et al., “Understanding your business through home-made simulator development,” Developments in Business Simulation and Experiential Learning, vol.26, pp.65–71, 1999.
[6] 中野健次，寺野隆雄，“意思決定の失敗事例をビジネスゲームで学ぶ—ケースとビジネスゲームの融合学習,” 情処学論，vol.49, no.10, pp.3354–3365, 2008.
[7] R.D. Duke, GAMING: the Future’s Language, Sage Publications, 1974.
[8] W.M. vanDer Aalst, A.H. Ter Hofstede, B. Kiepuszewski, and A.P. Barros, “Workflow patterns,” Distributed and Parallel Databases, vol.14, no.1, pp.5–51, 2003.
[9] T. Ishida, “*Q*: A scenario description language for interactive agents,” Computer, vol.35, no.11, pp.42–47, 2002.
[10] 林田智弘，西崎一郎，菅生雄矢，“エージェントベースシミュレーションを用いたムカデゲームにおける被験者の意思決定および学習に関する分析,” 日本オペレーションズ・リサーチ学会和文論文誌，vol.57, pp.27–43, 2014.
[11] W. Güth, R. Schmittberger, and B. Schwarze, “An experimental analysis of ultimatum bargaining,” J. Economic Behavior & Organization, vol.3, no.4, pp.367–388, 1982.
[12] 菱山玲子，中島　悠，“ゲーミングシミュレーションによる繰返し最後通牒モデルの動学的分析,” 第 24 回インテリジェント・システム・シンポジウム (FAN2014)，2014.
[13] 菱山玲子，“マルチエージェントシミュレーションにおけるゲーミングの利用,” 情報処理，vol.55, no.6, pp.557–562, 2014.
[14] 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,” 信学論（D），vol.J90-D, no.9, pp.2415–2422, Sept. 2007.
[15] 出口　弘，田沼英樹，市川　学，9.3.2 社会シミュレーション SOARS，ロボット情報学ハンドブック (松原仁，野田五十樹，松野文俊，稲見昌彦，大須賀公一編)，pp.616–626, ナノオプトニクス・エナジー，2010.

（平成 26 年 8 月 27 日受付，12 月 17 日再受付，
27 年 3 月 5 日早期公開）

中島　悠　（正員）

平成 18 年京都大学情報学修士課程修了．日本学術振興会特別研究員 (DC1)．平成 21 年同大学院情報学博士課程修了．博士 (情報学)．同大学特定研究員及び特定助教を経て，現在，東邦大学理学部情報科学科講師．都市上の人間行動のモデル化及びシミュレーションに興味をもつ．

菱山　玲子

京都大学情報学研究科社会情報学専攻修了．京都女子大学現代社会学部准教授を経て，平成 19 年早稲田大学大学院創造理工学研究科経営システム工学専攻准教授，現在同教授．人工知能，知識コミュニティデザインに関する研究に従事．博士 (情報学)．

中口　孝雄

平成 8 年京都コンピュータ学院鴨川校卒業．ATR 客員研究技術員，アントラッド主幹研究員を経て，平成 16 年京都情報大学院大学応用情報技術研究科修了．情報技術修士 (専門職)．NTT アドバンステクノロジを経て，現在，京都大学大学院情報学研究科特定研究員．サービスコンピューティングに興味をもつ．