社団法人　人工知能学会
Japanese Society of
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-DOCMAS-B102-4

# テーブル指向エージェントシミュレーションフレームワークの設計

## Designing Table-oriented Agent Simulation Framework

森 幹彦[1*]　津田侑[2]　喜多一[1]　上原哲太郎[3]
Mikihiko Mori[1], Yu Tsuda[1], Hajime Kita[1] and Tetsudaro Uehara[3]

[1] 京都大学学術情報メディアセンター
[1] Academic Center for Computing and Media Studies, Kyoto University
[2] 京都大学大学院情報学研究科
[2] Graduate School of Informatics, Kyoto University
[3] 特定非営利活動法人情報セキュリティ研究所
[3] Non Profit Organization The Research Institute of Information Security

**Abstract:** In this paper, authors present table-oriented agent simulation framework and indicate internal design of the framework. The table-oriented agent simulation is a concept of all data including agents' properties considered as cells of tables, and rules stored other spaces drive table-shaped data or agents. The framework for table-oriented agent simulation applies conventional thinking to create codes and tables, because of convenience to regard a name of a variable or a rule as the name of a file. The framework also provides a scaffold when a user executes the function of scaffolding with a template of a simulation process and initial data types. Finally, this paper shows an example on the framework to easily create a simulation program of Schelling's segregation model utilizing the template of Sugarscape model.

## 1　はじめに

社会現象を分析する手法としてエージェントシミュレーション（以下，ABS: Agent-Based Simulation と呼ぶ）が用いられている．ABS は，計算機上に人工社会を実現することにより，人や組織が行動主体となった社会プロセスをボトムアップに産み出してその秩序や規範の過程や結果を観察し分析する手法である．

ABS の実現のため，これまで様々なソフトウェアが提案されている．たとえば，市場経済 [1] や道路交通 [2] といった特定の領域に特化したソフトウェアや，領域を問わない汎用的なシミュレーションを実現するソフトウェア [3][4][5] がある．汎用性の高いシミュレーションソフトウェアは，社会現象を自由に記述できる一方で，自由度が高いがゆえにシミュレーション初心者にとってプログラミングの負担が大きくなった．一方で，特定の領域に特化するほど，その領域のシミュレーションを詳細に実行できるものの，他の領域への適用が難しいだけでなく利用に関するノウハウを転用できないという問題がある．

そこで本稿では，プログラミングの知識があるシミュレーション初心者を対象とした汎用的な ABS を実現するプログラミングフレームワークを提案し，その設計指針を示す．本フレームワークは，データ集合を扱うときに広く利用されているテーブルのアナロジーを組み込むことで，シミュレーション初心者への導入の敷居を低くできる．また，本フレームワークが提供するシミュレーションの枠組みを穴埋めしていきながらシミュレーションの設計と実装が行えるため，初心者のコーディング時間を減少させる効果を期待している．これにより，シミュレーション初期の試行錯誤段階において，実施のサイクルを短く，回数を多くでき，モデルの探索を容易にする．

*連絡先：京都大学学術情報メディアセンター
〒 606-8501 京都市左京区吉田二本松町

## 2　テーブル指向 ABS フレームワーク

本研究で提案する ABS のフレームワークは，ABS で用いられるあらゆるデータ集合をテーブルで表現する．テーブルはデータ集合を行列形式に並べるデータ構造で，コンピュータ科学の分野では一般的に広く用いられている．

エージェント等の行動を決定するルールは，テーブルで表現できるが，複雑なルールを表現するために無理に行列形式を利用する必要はなく，行動スクリプトと

してプログラムコードで記述できる．同様に，ABS で必要とされるエージェントの行動を制御するスケジューリング機構が必要であるが，スケジュールスクリプトとしてプログラムコードで記述できる．これらのプログラムコードは，ライブラリやテンプレートを含む支援コードの提供により容易な構築環境を提供する．

また，本フレームワークは，ABS のひな形をテンプレートや実データを用いて作成するスキャッフォールディング機能を備える．このとき，ファイル名やテーブル名，変数名，関数名に命名規則を用意し，そのような規約にしたがうことで自動的な作成の範囲を広くできる．

### 2.1 フレームワークで用いられるテーブル

提案する ABS フレームワークでは，構築する社会シミュレーションにおける人工社会をエージェント，関係，環境の 3 つのテーブルで表現する（図 1）．ここで，各テーブルは，1 つの ABS に対して複数用意することができる．

**エージェントテーブル**　エージェントテーブルは，エージェントの個々の特性を表したもので，エージェント数を $m$ 体，エージェントが持つ特性数（属性数）を $n$ 個とした $m \times n$ 行列で表される．もし，属性数が異なる別種のエージェントを用意する場合には，2 つの方法が考えられる．1 つは，各種ごとの属性集合の和集合を列とし，空要素を許すよう実装する方法である．もう 1 つは，エージェント種別ごとに別テーブルを用意し，行動テーブルで融合する方法である．本フレームワークではどちらの表現も ABS の実装次第としている．

**関係テーブル**　関係テーブルは，エージェント間の接続関係をグラフ理論などで用いられる隣接行列で表したものである．実現する人工社会のモデルによってエージェント間の接続関係の有向・無向，重み付けの有無を自由に選択し記述できる．図 1 では，エージェントの接続関係は無向であり，それぞれの接続に重み付けがある場合を示している．

**環境テーブル**　環境テーブルは，エージェントとは独立したデータ集合で，人工社会自身の環境情報の一覧をテーブルで表現したものである．各エージェントは環境テーブルから人工社会の環境情報を参照し行動する．たとえば，環境テーブルとして，格子モデルにおける各セルの配置にもとづくテーブルや，ネットワークモデルにおける各スポットの関係を隣接行列で表現したテーブルが考えられる．このように，環境は ABS の対象に寄って様々であることから，適切なテーブルを採用する必要がある．

### 2.2 その他のデータ

2.1 節で述べたフレームワークで用いるテーブルの他に，ABS ではエージェント間で共有するデータやテーブルが必要になる．そこで，後に 3.2 節で述べる環境オブジェクト内に属性変数やテーブルとして用意し参照できるようにする．

### 2.3 行動ルールとスケジューリング機構

定義された行動ルールは互いに独立で，複数の行動ルール間ではエージェントの行動は互いに影響しない．一方で行動ルール内ではエージェントの状態変化に差が出ないように行動を定義することが求められる．

スケジューリング機構は，エージェントごとの行動ルールを逐次実行する．それが繰り返されることによってシミュレーションが実現される．このようなスケジューリング自体がルールと言えることから，以降ではエージェントの行動ルールとスケジューリングを含めてルールと呼ぶことにする．

### 2.4 スキャッフォールディング機能

ABS を構築する際，典型的なエージェントモデルがフレームワークから提供されると，利用者がプログラミングにかける手間を低減させることができる．スキャッフォールディング機能を利用することで ABS の「土台」が作成される．具体的には，エージェントの状態判定や環境状態のチェックといった基本的な行動ルールが提供される．作成された土台は，フレームワークの利用者がその後に修正して利用することを前提にしたものである．これにより，プログラミングの自由度を確保する．このような実行の流れを図 2 に模式的に示す．初期状態では，何もないところにスキャッフォールディング機能によって，枠組みとなる土台と必要な部品であるデータ（エージェントと環境要素）やルール（矢印）が用意される．これらをもとにして利用者の求める ABS を構築していくことになる．

## 3 フレームワークの内部設計

本フレームワークは，統計処理向けプログラム言語である R 言語[1]による実装を想定していることから，内部設計においても R 言語の言語的特徴と制約を反映し

[1] http://www.r-project.org/

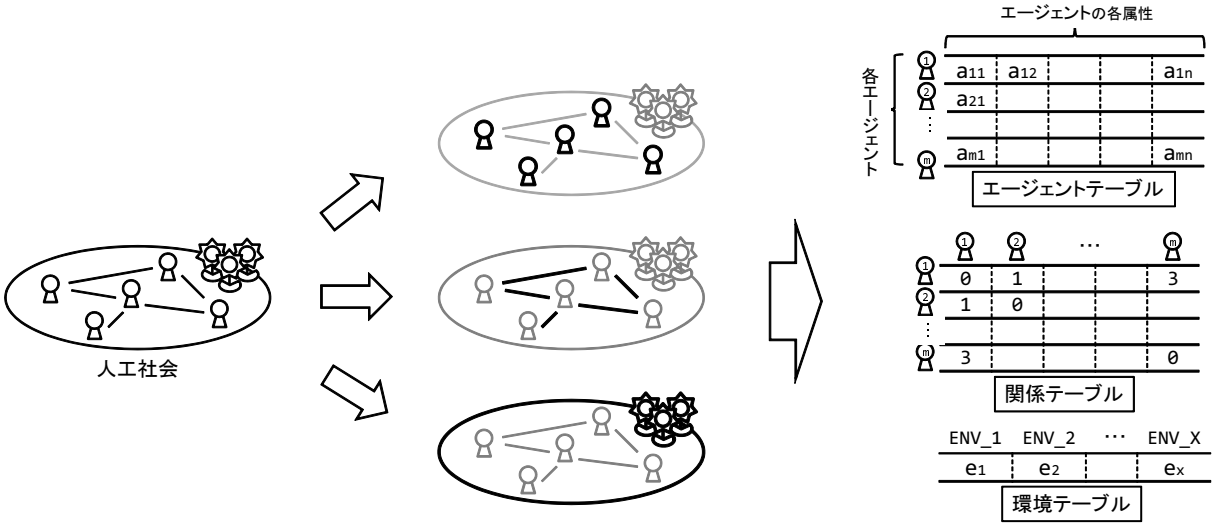


図 1: テーブル指向 ABS フレームワークで用いられるテーブル

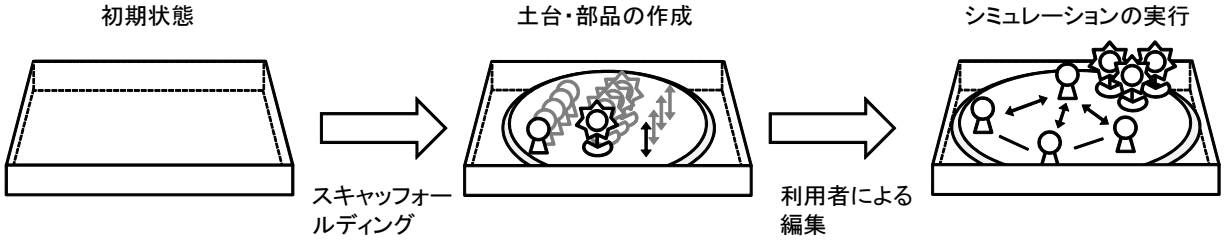


図 2: スキャッフォールディングによる ABS 構築の流れの模式図

た作りになっている．R 言語は，言語の基本処理にテーブルの操作が用意された軽量言語としての記述容易性がある．また，統計処理言語として統計処理に関するライブラリやグラフ等の可視化ライブラリが豊富に用意されているため，ABS の実行結果を同じプログラムコード中にシームレスに埋め込むことが可能である．

### 3.1 命名規則とファイル配置に関する規約

本フレームワークは，ファイルやデータ等を半自動的に生成するために，命名規則とファイル配置に関して規約を設けている．これにより，データ等とそれを補完するファイルの間，ファイルとファイルの内容に関する記述の間で自動的な対応がとれるようになっている．これにより，スキャッフォールディング機能が利用者に求める情報を少なくできる．また，利用者が構築する場合にも，この規約を通じて容易にフレームワークを利用できる．規約の詳細は以降の節で個々に述べる．

### 3.2 データとルールの配置

各種データは，オブジェクトと呼ぶ大枠の入れ物にテーブルまたは属性変数，ルールとして格納される．そのためのオブジェクトとして，エージェントテーブルを格納するエージェントオブジェクト agents と，関係テーブル，環境テーブル，その他の属性をまとめた環境オブジェクト worlds，エージェントの行動ルールとスケジュール機構をまとめたルールオブジェクト rules の 3 つのオブジェクトを用意した．これにより，例えばエージェントテーブルが複数ある場合にも，一体となった変数として利用が可能になる．

このような変数は，R 言語のオブジェクト指向プログラミングの枠組みで実装されている．そのため，データだけでなくデータを操作するメソッドを変数内に定義できる．さらに，データ以外にルールのようなプログラムコードもメソッド（関数）として無理なく収められる．

## 3.3 ファイルの配置

これらの変数の初期値は，ファイル群として保存されていると想定されている．そのため，各変数を所定のファイルパスのディレクトリにおける同名のサブディレクトリとして扱う．この変数名ディレクトリ内には，テーブルや属性変数，ルールがそれぞれファイルとして納められている．テーブルは個々にファイルが用意され，デフォルトで CSV 形式で保存される．ファイル名は，テーブル名と同一である．また，属性変数は，一括で 1 つのファイルにまとめて保存され，デフォルトで CSV 形式である．このときのファイル名は attributes.csv である．同様に，ルール等のメソッド群も一括で rules.r ファイルに R スクリプト形式で保存される．

## 3.4 支援コード

本フレームワークは，標準的なエージェントの行動と行動順を想定し，スキャッフォールディング機能による土台としてシミュレーションに関わるプログラムコードを出力できる．これにより，初期の各種テーブルや変数，ルールを独自に検討する手間の省力化につながる．また，シミュレータに隠蔽されることなく ABS の挙動を把握するための追跡が利用者の手の中で行え，すべてのコードを修正できる．

このような土台コードを出力するために，スキャッフォールディング機能を提供するライブラリと，各シミュレーション種別に適した最低限のコードを出力可能にするテンプレートを提供する．

### 3.4.1 オブジェクトの種別と骨組み

本フレームワークは，標準的な ABS におけるエージェントの行動として，観測・判断・実施の 3 つの行為があると想定し，それぞれの行為について各エージェント一括で実行するメソッド observe, decide, execute を用意している．また，判断と実施の間には，エージェント間で干渉し合うことが想像できることから，システム側から確認・調停の 2 つの行為をメソッド validate, mediate として提供され，実施前に最終的な実施内容の調整が行えるようにする．したがって，標準的なスケジューリング機構として，観測・判断・確認・調停・実施が一列に並んだものとして提供される．

上述のメソッド群には，各メソッドを呼び出す前後に before.*と after.*（‘*’ にはメソッド名が入る）のフックメソッドも用意されている[2]．これらのメソッドは，各行為前の整形やフィルタ，各行為後のロギング等を想定している．

シミュレーション状態の保存する関数 save.sim も用意されている．これは，3.3 節で述べたファイル配置にもとづいて初期ファイルの場所に上書きするか，別のディレクトリに保存するかを指定可能である．また，保存したデータを load.sim 関数が復元したところから始める continue.sim 関数も用意している．ここで，load.sim 関数は，読み込むデータが初期データなのか復元目的のデータなのかを区別しない作りになっている．したがって，シミュレーションの開始時に初期データを読み込む場合にも利用可能である．

この他に，表の各行の要素を関数名と引数として扱うための支援関数[3]や，途中経過のデータを 1 つのファイルに追記してロギングする log 関数と各オブジェクトの log メソッドが用意されている．

### 3.4.2 スキャッフォールディング機能とテンプレート

3.4.1 節で述べた骨組みの初期構築のために，スキャッフォールディング機能を提供する．このとき，ABS の種別ごとに用意されたテンプレートを利用することにより，スキャッフォールディングが作成する土台にあらかじめ種別ごとに分かっているコードを自動生成する．

まず，スキャッフォールディング機能では，初期ファイルを用意する．初期ファイルは，作成する ABS に必要なデータをフレームワークに示して土台作りの基本にさせる．このとき，3.3 節で述べたディレクトリ構成とデータファイルといったシミュレーションに利用するデータのほかに，スキャッフォールディング機能にデータの意図を伝えるための記述ファイルを用意する．ファイル名は，3.3 節で述べた規約にしたがったファイル名に-desc.csv を付けた CSV ファイルとしている．たとえば，エージェントテーブルであれば，agents ディレクトリ下の agents.csv（デフォルト）に対する記述ファイルとして agents-desc.csv を agents ディレクトリ下に配置する．このような記述ファイルは，テーブルであれば各列項目，属性変数であれば変数ごとに対して，名称と変数型等を記述する．

次に，scaffold 関数にテンプレート名や初期ファイル，作成ファイルの保存先を指定し，実行するとテンプレートにしたがって 3.4.1 節で述べたオブジェクトやルールが初期ファイルと照合されながら保存先ディレクトリに作成される．

その後に，保存先ディレクトリの各ファイルを適宜，利用者の想定する ABS の完成に向けて修正していく．そして，run.sim 関数を呼び出してシミュレーションを実行する．途中経過は，3.4.1 節で述べたように log 関

[2] R 言語における ‘.’ はインスタンスのメソッド呼び出しを意味せず，Java や C 言語でいうところの ‘_’ 程度の変数名を構成する文字にすぎない．

[3] ただし，これは R 言語に内蔵されている関数を “薄く” ラッピングする関数である．

ソースコード 1: 基本的な Schelling の分居モデルの ABS の構築

```
1  # 提案フレームワークの読込み
2  library("ftabas")
3
4  sim <- scaffold(template="sugarscape")
5  save.sim(sim, path="./schelling")
6
7  # ここで，./schelling/以下に展開された
8  # ファイルを適宜修正する
9
10 sim <- load.sim(path="./schelling")
11 run.sim(sim, options=some.options)
```

数でロギングできるほか，save.sim 関数でオブジェクトを保存し，load.sim 関数と continue.sim 関数により復元と続行が可能である．

テンプレートは，骨組みだけのものが用意されている．また，このテンプレートには，典型的なモデルを指定でき，あらかじめ用意された各オブジェクトの実体が含まれる．その他に，以前に作成した ABS に記述ファイルを用意することでテンプレート化することもできる．

# 4 テーブル指向 ABS フレームワークの適用例

本節では，本提案フレームワークを適用した ABS の作成例を示す．ソースコード 1 は，Sugarscape モデルをテンプレートとし，提案フレームワークでシェリングの分居モデルを実装する過程を概略で表している．本例では，Sugarscape モデルとシェリングの分居モデルが格子空間におけるエージェントの移動という点で共通すると考え，環境テーブルが流用できると想像してこのような実行過程を示している．

まず，scaffold 関数を利用することで ABS の土台が作成される．ここでは，sugarscape テンプレートを指定している．作られたオブジェクトは sim 変数に保存される．R 言語ではインタラクティブなコマンドシェルが標準で提供されていることから，このまま sim 変数に対して様々な変更をシェル上で実施することも可能である．その場合には，適当な場面で save.sim 関数を呼び出せばよい．本例では，別途用意するエディタ等でファイルを直接編集する場合を示している．

モデルの修正が完了した後に，load.sim 関数により呼び出して，sim 変数に最新のオブジェクトを読み込む．最後に，run.sim 関数を実行することで，ルールオブジェクトに保管された行動ルールとスケジューリング機構が呼び出され実行される．

# 5 関連研究

汎用的な ABS を構築するためのプログラムライブラリに Swarm がある [3]．Swarm はエージェントの作成やスケジュールリング，GUI や可視化のためのライブラリを提供し，利用者は Java か Objective-C により ABS を実装する．同様ものとして，MASON[5] がある．MASON は Java で記述されたライブラリで，シミュレーションの実行スケジュールやエージェントの状態の管理とシミュレーションの途中経過の可視化ツールが提供される．このようなシステムを利用すると，シミュレーションの内容を自由に記述でき，汎用的なシミュレーションを構築できるが，一方でそのようなシミュレーションを構築する場合，利用者には高いプログラミング能力とともにシミュレーションのために必要な能力も要求される．

プログラミング初心者でもシミュレーションができるシステムとして，PlatBox[6] がある．PlatBox は，UML を用いて GUI 上でモデリングすることによりシミュレーションを実行できる．同様に GUI 上でモデリングするシミュレータとして SOARS[7] がある．SOARS は，エージェントの振る舞いを役割ルールとスポットルールで実行するところに特徴がある．これらのルールは，テーブル上に所定の書式で記述していく．また，シミュレーションの実行結果を可視化することもできる．これらのシステムは，プログラミング初心者には受け入れやすいものであるが，深いところに手を入れようとすると Java によるコーディングが必要になってしまう．

ライブラリ志向と GUI 構築志向の両方を持つシステムとして，Repast[4] がある．Repast は，エージェントの振る舞いをフローチャートとして GUI 上でパーツを組み合わせることでモデリングすることができる．さらに Repast は，フローチャート記法とは別に，ReLogo と呼ばれる Logo のタートルを取り入れた groovy プログラムでモデルを記述することも，Java プログラムとして記述することもできる．これらのモデルを使ったシミュレーションの経過は付属のライブラリで可視化できる．

artisoc[8] は，エージェント等の振る舞いの記述に特化した記法提供して GUI と併用によりシミュレーションを行うシステムである．専用のライブラリを利用した BASIC に似た記法によりエージェント等の振る舞いを記述でき，簡便に実行できる．artisoc は，システムが提供する枠組みの中で ABS を実装することになる．

ますめ [9] は，シミュレーションを情報教育に用いるためのプログラミング環境として提案されている．シートエリアにプログラムまたは値を入力していくことでプログラムを作成できる．ますめや SOARS の記述法は，本提案のフレームワークにおけるテーブル志向による記述に近い発想である．しかし，ますめや SOARS は

GUI 上で完結させようとしているが，本フレームワークでは R 言語での実行可能なデータとして扱う点で異なる．

GUI によるモデリングは，視覚的にわかりやすく，直観的に ABS を構築する方法ではあるが，GUI 操作の煩雑さを伴う．また，プラットフォームシステムを抜け出すことが難しく，細かい分析のためには別のアプリケーションと併用する必要がある．一方で，ライブラリのみの提供では，プログラミング初心者への敷居が高い．そこで，本研究の提案するフレームワークでは，規約に基づく初歩的な ABS をスクリプト記述により容易に構築できるようにした．また，軽量言語の持つ学習の容易さによりプログラミング初心者がプログラミングの基礎知識（プログラミング言語の構文がわかる程度の知識）を得るまでに時間をかけなくて済む．

## 6 おわりに

本稿では，基礎的なプログラミング知識のあるシミュレーション初心者を対象とした ABS フレームワークを提案し，その設計について述べた．シミュレーション初心者への導入の敷居を低くするためにテーブルのアナロジーを組み込んだ．利用者はフレームワークが提供するスキャッフォールディング機能を用いて ABS のひな形から土台を作成し，人工社会におけるエージェント，関係，環境の 3 つのテーブルとルールを記述することで ABS を実現できる．この土台は，ABS のすべてのコードを含むため，すべての面で修正可能であるが，テーブルの構造やルールを大きく変えない場合には，修正箇所を少なくして ABS の構築が可能である．

現在，本提案フレームワークの実装途中であり，基本的な動作の検証ができている．今後は，安定的な動作を目指すだけでなく，各種のデータに頑健に対応できるようにする．また，シミュレーション結果の可視化のため，R 言語の用意する可視化ライブラリへの橋渡しを実装する予定である．

さらに，テンプレートを充実させるため，Sugarscape モデル，シェリングの分居モデル，繰り返し囚人のジレンマ，捕食者-被食者モデルといった古典的なモデルのテンプレートを用意する．最後に，これらのテンプレートを用いた様々な ABS を本フレームワークで構築し，記述性の評価と実行性能の評価を行う．

## 参考文献


[1] 喜多一, 森直樹, 小野功, 佐藤浩, 小山友介, 秋元圭人. 人工市場で学ぶマーケットメカニズム —U-Mart 工学編—. 共立出版, 2009.

[2] M. Balmer, K. Meister, M. Rieser, K. Nagel, and K.W. Axhausen. Agent-based simulation of travel demand : Structure and computational performance of MATSim-T. In *the 2nd TRB Conference on Innovations in Travel Modeling*, 2008.

[3] Nelson Minar, Rogert Burkhart, Chris Langton, and Manor Askenazi. The swarm simulation system: A toolkit for building multi-agent simulations. Working Papers 96-06-042, Santa Fe Institute, June 1996.

[4] N. Collier. RePast: An Extensible Framework for Agent Simulation. *The University of Chicago's Social Science Research*, 2003.

[5] S. Luke, C.R. Claudio, P. Liviu, S. Keith, and B. Gabriel. MASON: A Multiagent Simulation Environment. *SIMULATION*, Vol. 81, No. 7, pp. 517–527, 2005.

[6] 井庭崇, 中鉢欣秀, 松澤芳昭, 海保研, 武藤佳恭. Boxed Economy Foundation Model:社会・経済のエージェントベースモデリングのためのフレームワーク. 情報処理学会論文誌, Vol. 44, No. SIG 14, pp. 20–30, 2003.

[7] 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. 電子情報通信学会論文誌, Vol. J90-D, No. 9, pp. 2415–2422, 2007.

[8] 山影進. 人工社会構築指南—artisoc によるマルチエージェント・シミュレーション入門—. 書籍工房早山, 2007.

[9] 荻野哲男, 藤岡健史, 柳瀬大輔. 教育現場での実践に向けたプログラミング実行環境「ますめ」の試作. 情報処理学会研究報告, Vol. 2011-CE-111, No. 5, pp. 1–6, 2011.