知能と複雑系　123－15
(2001. 1. 10)

# Boxed Economy 基礎モデルのプロトタイピング: デザインパターンによるアプローチ

中鉢欣秀 †1†2, 井庭崇 †1†3†4, 松澤芳昭 †1†2
浅加浩太郎 †1†2, 海保研 †2†6, 廣兼賢治 †5, 高部陽平 †6

†1 慶應義塾大学 政策・メディア研究科　†4 フジタ未来経営研究所 リサーチアソシエイト
〒 252-0816 神奈川県藤沢市遠藤 5322　†5 慶應義塾大学 環境情報学部
†2 合資会社ニューメリック　†6 慶應義塾大学 SFC 研究所 訪問研究員
†3 日本学術振興会 特別研究員
*architects@boxed-economy.org*
*http://www.boxed-economy.org/*

あらまし

エージェントベース経済シミュレーションの共有基盤である「Boxed Economy シミュレーションプラットフォーム」の構築に向け、プロトタイプ実装を行なった。実装に際し、「Boxed Economy 基礎モデル」にデザインパターンを適用し、柔軟性・再利用性の高いコンポーネントフレームワークとするための設計を試みた。その結果、エージェントの動的な振舞の実現と、概念モデルと実装モデルを関係付ける手法としてデザインパターンの利用が有用であることが示された。

キーワード　オブジェクト指向モデリング, エージェントベース経済シミュレーション, デザインパターン, UML, Java, JavaBeans

# Prototyping of Boxed Economy Foundation Model: Approaching with Design Patterns

Yoshihide Chubachi†1†2, Takashi Iba†1†3†4, Yoshiaki Matsuzawa†1†2
Kotaro Asaka†1†2, Ken Kaiho†2†6, Masaharu Hirokane†5, Yohei Takabe†6

†1 Graduate School of Media and Governance, Keio Univ.　†4 Research Associate of Fujita Institute of Future Management Research
5322 Endo Fujisawa-city Kanagawa, 252-0816, Japan
†2 Numeric & Co., Ltd.　†5 Dept. of Environmental Information, Keio Univ.
†3 JSPS Research Fellow　†6 Visiting Researcher of SFC Institute, Keio Univ.
*architects@boxed-economy.org*
*http://www.boxed-economy.org/*

Abstract

We developed a prototype of a foundation framework named the "Boxed Economy Simulation Platform" which is a platform of agent-based economic simulations. Design patterns are applied to "Boxed Economy Foundation Model" to build a flexible and reuseful component framework. It showed that using design patterns are an effective method to implement dynamic act of agents and to connect between conceptual model and implementation model.

key words　object-oriented modeling, agent-based economic simulation, design pattern, UML, Java, JavaBeans

# 1 はじめに

従来のエージェントベース社会シミュレーション研究においては、研究対象に特化したシミュレーションの開発が行なわれてきた。そこでは新たなシミュレーションを開発する場合に、毎回白紙の状態からモデル化と実装が行なわれている。また、実装の詳細な仕様(特に、ソフトウェアのソースコード)が公開されていない研究例も多く、他の研究者による拡張や再利用が困難となっている。これらは、社会シミュレーション開発の方法論としてソフトウェア工学的見地からも改善されるべき問題である[1][2][3]。特定の事例を対象として社会シミュレーションが開発されるだけではなく、再利用可能な「フレームワーク」が提供されるならば、今後の本分野における研究の発展に寄与すると思われる。

このような現状に対し、私達はエージェントベース経済シミュレーションの共有基盤として「Boxed Economy」というシミュレーションプラットフォームを提案し、その構築に取り組んでいる。Boxed Economyは経済シミュレーションにおけるフレームワーク、およびそこで利用可能な部分モデルコンポーネントを作成・蓄積するための基盤を提供する[4][5]。本論文では、社会モデルにデザインパターンを適用することによって柔軟性・再利用性の高いコンポーネントフレームワークを設計することの検討と、これに基づくプロトタイピングの現状について報告する[1]。

# 2 基礎モデルとデザインパターン

Boxed Economyでは、エージェントベースの経済シュミレーションのための基本的な社会モデルとして「Boxed Economy基礎モデル」(以下、基礎モデル)を提案している。基礎モデルの詳細については、論文[6]を参照されたい。今回、この基礎モデルに対して以下の2つの観点から「デザインパターン」[7][8]の適用を検討した。

- エージェントの動的かつ柔軟な振舞を実現する
- 基礎モデルと実装モデルを疎に結合させ再利用性を確保する

[1] 本論文で行なう議論はエージェントベース**経済**シミュレーションであるが、本論文の内容は**市場**シミュレーションや**政治**シミュレーションなど、エージェントベース**社会**シミュレーション一般に適用可能である。

エージェントベース社会シミュレーションにおいて、エージェントの行動、他のエージェントの関係などはシミュレーションの実行中に動的に変化させなくてはならない。また、抽象的・概念的な「社会モデル」をソフトウェアとして動作させるためには、基礎モデルに対して具体的な「実装モデル」の追加が必要となる。このとき、再利用性を高めるためには社会モデルと実装モデルの関係ができるだけ分離されていることが望ましい。これらを実現するために、デザインパターンから主にオブジェクト間の柔軟な関係を構築する目的のものを抽出し、利用の可能性について検討を行なった。

## 2.1 Observerパターン

Observerパターン[7]は、オブジェクト間の依存関係を動的に構築することを可能とする。基礎モデルでは、シミュレーションの進行に伴い、オブジェクト間の関係が動的に変化する。例えば、経済主体と経路は互いに依存関係を持つが、その関係が構築されている時間は非常に短く、財の移動が必要な場面において経路の開設と閉鎖が繰り返される。これにObserverパターンを適用すると、クラス間の関係を疎に保ちながら、必要に応じて関係を変化させることが可能となる。

また、シミュレーション中、経済主体の行動を外部から観察する必要がある。これに対するObserverパターンの適用は、最も典型的な利用方法である。

## 2.2 Delegationパターン

Delegationパターン[8]は、機能を提供するもとのクラスに対して機能追加を行なうときに、継承ではなく、別なクラスに機能を委譲することにより、クラスを拡張する方法である。経済主体は、シミュレーションの実行時においてその機能を動的に変化させる必要があるが、静的な継承によるサブクラスでの機能実現では、これに対応できない。そこで、経済主体の機能(行動・思考)を拡張するための手段として、Delegationパターンを用いることにする。

## 2.3 State/Strategyパターン

State/Strategyパターン[7]はアルゴリズムの種類をオブジェクトごとに選択したり、動的に変化させるための仕組みである。基礎モデルに対して、経済主体の振舞の実装にState/Strategyパターンを利用す

る。シミュレーションにおいて、経済主体は自ら行動(Strategy)を決定する。行動の詳細は、状態(State)に分割し、他の経済主体とのインタラクションによる状態変化として記述することができる。このことにより、基礎モデルをもとにしたフレームワークに対して、経済主体の機能を様々に拡張可能とし、他の経済主体との経済行為の詳細を実装することができる。

### 2.4 Compositeパターン

Compositeパターン[7]は、ツリー状に構成された全てのオブジェクトに対して、共通のインターフェースを持たせることにより、ツリー上のオブジェクトを一貫した方法で操作できるようにする。基礎モデルにおいては、社会集団に複数の個人が属しているというような構造を実現する場合に、このCompositeパターンを適用する。

### 2.5 Visitorパターン

Visitorパターン[7]は、Compositeパターンで構成された全てのオブジェクトに対して統一したアクセス手法を提供する。これを用いて、シミュレーションに存在する全ての経済主体をツリー構造に従って走査することが可能となる。例えば、ある社会集団に属する全ての経済主体が所有する財について、何らかの統計処理を行ないたい場合などに使用することができる。

## 3 プロトタイピング

今回のプロトタイプ実装は、前節で検討したデザインパターンの基礎モデルへの適用の妥当性を確かめることを主眼とした。実装のための環境としてJava言語を用い、実装モデルとしてJavaBeansコンポーネントフレームワーク[9]を使用した。プロトタイピングの目的は、基礎モデルに従い、デザインパターンを用いてエージェントの動的な振舞を実装することと、基礎モデルに一切の変更を加えることなく基礎モデルと実装モデルとを疎に結合させ、実際に動作するソフトウェアを作成することである。なお、今回のプロトタイピングにおいて、エージェントの振舞についてはあらかじめ定められたシナリオに基づき経済的行動を行なうように実装した。

### 3.1 Java言語による実装モデル

プロトタイピングで使用したJava言語は、現在、広く普及しているオブジェクト指向のプログラミング言語であり、言語の機能としてソフトウェアの部品化(コンポーネント化)が可能である。加えて、経済シミュレーションのように本質的に計算量を必要とするアプリケーションへの利用に際して不安要素となる実行速度についても、実行時コンパイラやHotSpot[10]等の技術の登場による改善が見られる。

今回は、JavaBeansフレームワークを「実装モデル」としてコンポーネント化を行なった。JavaBeansはJava言語によるコンポーネントフレームワークである。この技術はGUIコンポーネントを作成することに広く利用されており、JavaのGUIコンポーネントパッケージである「Swing」においても用いられているものである。本プロトタイプでは、基礎モデルにあるクラスを、JavaBeansコンポーネント、あるいはSwingコンポーネントの子クラスとして実装する[2]。これにより、プロトタイプ実装でありながらも、GUIを備えた訴求力のあるシミュレータを、短時間で完成させることができる。

また、JavaBeansフレームワークには、随所にデザインパターンが取り入れられている。このことも今回のプロトタイプ実装において同技術を利用するメリットであり、前節で検討した基礎モデルにデザインパターンを適用することで実装モデルと結合するアプローチにも適している。

### 3.2 プロトタイプ実装

今回実装したプロトタイプ版シミュレータの実行画面を図1に示す。シミュレータは、Javaアプリケーションとして作成され、1つのメインフレームを持つ(図1、右上)。このフレームには、シミュレーションの初期設定を行なうための「世界」コンポーネントが配置されている[3]。

また、経済主体の行動を監視するための「経済主体監視Frame」を表示させることができる。ここには、エージェントの行動、エージェントが所有する財の変化などが表示される(図1において左側上下に配置されているフレーム)。

[2] 将来的には、基礎モデルクラスはSwingに依存しないクラスとし、より実装モデルと切り離されることが望ましい。

[3] このコンポーネントは今回のプロトタイプ実装のために基礎モデルとは別に追加したものである。

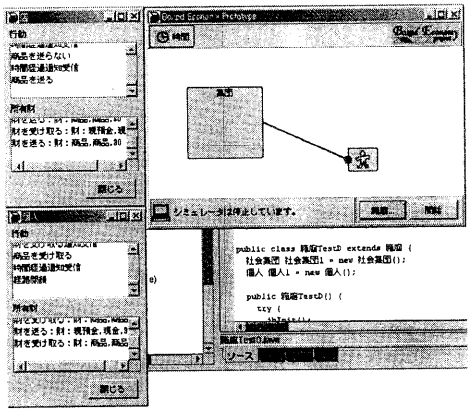

図 1: プロトタイプ実行画面

### 3.3 世界コンポーネントと時間コンポーネント

世界コンポーネントには、「時間」コンポーネントが配置されている (図 1、メインフレーム内左上)。同様に、シミュレータの動作を制御するためのボタン、メッセージ等の表示のためのラベルを用意した。

世界コンポーネントの中央には基礎モデルの「空間」コンポーネントが配置されている。この空間コンポーネント上に、基礎モデルの「経済主体」及びその子クラスである「社会集団」と「個人」が配置され、これがシミュレーションの初期状態となる。

世界コンポーネントは初期化の際、空間コンポーネントに配置されている全ての経済主体及びその子クラス (以下、単に「経済主体」と言った場合、その子クラスも含む) に対して、時間コンポーネントより通知される「時間 Event」が受信されるように設定する。時間コンポーネントは、シミュレーションが動作中、一定時間ごとに時間 Event を発行し、経済主体がこれを受信することで、経済主体に行動の機会を与える。

本プロトタイプ実装では、世界コンポーネントは経済主体の行動を監視するための「行動監視 Event」、「所有財監視 Event」、「経路監視 Event」を受信する。これらは経済主体より通知され、シミュレーションにおける経済主体の行動をユーザが監視するために使用する。これらのイベントにより、画面上の経済主体をマウスでクリックすると、「経済主体監視 Frame」が表示されて、経済主体の行動を監視することができる。なお、以上の仕組みは、デザインパターンの Observer パターンを利用している。

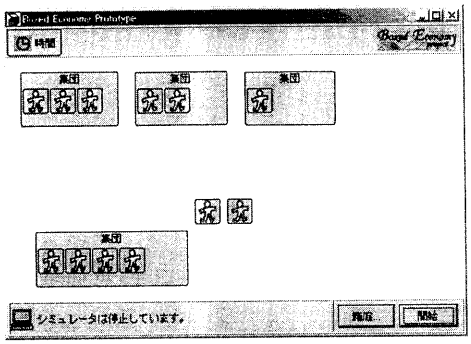

図 2: 社会集団と個人コンポーネント

### 3.4 経済主体コンポーネント

「経済主体」は、Boxed Economy シミュレーションモデルにおけるエージェントの実装クラスであり、「社会集団」と「個人」を子クラスにもつ。主な役割として、「財」及び「情報」の所有、Delegation パターンによる「機能 (思考・行動)」の実行、「関係」の保持と「経路」の開設、「情報」の記憶などをつかさどる。

「社会集団」は、他の経済主体を内包しながら、自律した経済主体としての機能を実現するものであり、Composite パターンを利用している。また、経済主体は、複数の社会集団に同時に所属することもできる。

「個人」は、「欲求」コンポーネントを所有する経済主体である。欲求は個人を刺激し、個人の行動を駆動する役割を担う。

なお、本プロトタイプでは、経済主体にあらかじめ用意されたシナリオに基づく経済活動を行なわせることを目標として作成された。従って、エージェントが外部・内部の環境を認知し、自律的に意思決定を行なう仕組みの実装は行なっていない。基礎モデル中の、「記憶」コンポーネントと個人コンポーネントが所有する「欲求」コンポーネント、及び財に付随する「情報」コンポーネントは、エージェントの意思決定に使用されるものであるため、今回のプロトタイプでは実装を省略した。

本プロトタイプにおいて「社会集団」と「個人」の経済主体を配置した画面を図 2 に示す。

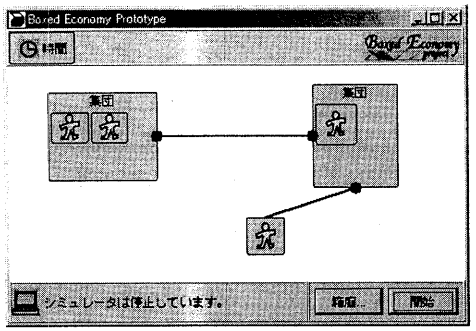


図 3: 関係と経路コンポーネント

### 3.5 所有財・財コンポーネント

「所有財」コンポーネントは、経済主体が所有する財の管理を行なうクラスである。財の追加や削除は本クラスのメソッドとして実装されている。

「財」の属性として、基礎モデルにある「種類」と「名称」、「数量」を実装した。ここで、基礎モデルの考え方では「価格」は財の属性として含まず、経済主体の状態や、取引をする他の経済主体との関係によって決定されることになっている。しかし、今回のプロトタイプでは簡単のため数量と価格が等しいとした。例えば、100 万円分の商品は、数量が 100 万であるとして表す。また、財は全て分割可能であるとした。例えば、「1 万円の現金」という財を分割して、「2 千円の現金」と「8 千円の現金」に分け、その一方を他の経済主体に経路を経由して送ることができる。

### 3.6 関係・経路コンポーネント

「関係」・「経路」についても、グラフィカルに表示できる (図 3)。図では個人・社会集団間に関係が構築され、その上で経路が開設された状態を示す。関係及び経路は、片方向であるため、相手先を黒丸で示すことにした。

「関係」及び「経路」には Observer パターンを実装している。関係のある他の経済主体との間で経路を開設したい場合、経済主体は「関係」に対して「関係 Event」を発行し、「関係」コンポーネントはこれを監視している。関係 Event には、相手先が含まれることから、これをもとに関係は経路を構築する。同様に、「経路」コンポーネントも「経路 Event」を監視している。経路に財を送りたい場合、経済主体

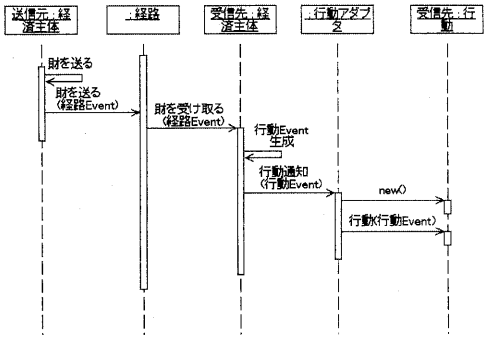


図 4: 経路 Event が引き起こす行動

は同 Event を発行する。これにより経路は、自分と接続されている経済主体に対して財の到着を知らせることができる。この経路 Event は経済主体により行動 Event に変換され、経済主体に行動の機会を与える (図 4) [4]。

### 3.7 「機能 (思考・行動)」コンポーネント

本プロトタイプにおいては、エージェントの思考機能の実装は行なっていない。従って、「機能 (思考・行動)」コンポーネントを単に「行動」コンポーネントとして実装した。行動コンポーネントは、経済主体の一連の行動 (Strategy) を決定する「行動アダプタ」と、状態遷移により行動を実現する「行動」の 2 つのクラスに分離されている。行動アダプタは、行動の決定と、行動への「行動 Event」の伝達 (Delegation) を行なう。行動は、イベントによって駆動される状態機械 (オートマトン) であり、実装には State パターンが利用されている。

これをもとにして、2 つの経済主体が経済活動を行なう際のシーケンス図を図 5 に、双方の行動の状態遷移を図 6 に示す。

## 4 今後の発展に向けて

今回のプロトタイプ実装により、社会モデル (Boxed Economy 基礎モデル) と実装モデル (Java Beans コンポーネントフレームワーク) をデザイン

[4] 本図は UML[11] で規定されている「シーケンス図」による表現である。Boxed Economy では、UML を用いて基礎モデルのデザインを行なっており、研究者間での共通言語となっている。

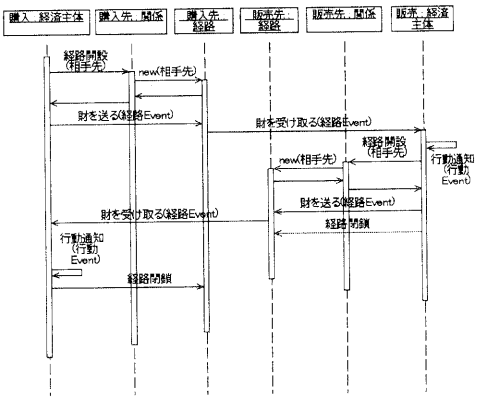


図 5: 購入・販売シーケンス図

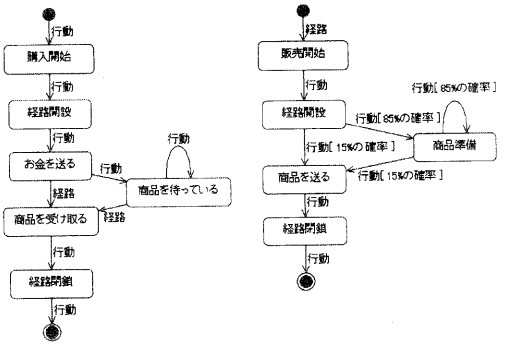


図 6: 購入・販売ステートチャート

パターンを用いて関連させる手法の有用性を確認することができた。経済主体エージェントの振舞を、エージェント間のインタラクションによって動的に変化させることができ、基礎モデルの構造を何ら変更することなく、実際に動作するソフトウェアが実装されている。このことから、デザインパターンの利用が、拡張性・柔軟性を備えたエージェントベース社会シミュレーションのプラットフォームを構築するため有効な方法であるといえる。

また、本研究における、基礎モデルのデザインからプロトタイプの実装に至る過程は、今後、社会シミュレーションを開発する一つの手法として提案できると考えている。例えば、社会モデルの設計作業と、実装モデルに関する事柄とを完全に分離することは、コンピュータ技術についての知識の多寡を問題にすることなく、様々な研究者のモデル設計作業への参加につながり得る。また、オブジェクト指向のモデル表記法として定着している UML(Unified Modeling Language) を研究者間の共通言語として使用することによって、非常に円滑なコラボレーティブワークが行なえることも、私達の経験として得られている。これらについては、機会を改めて紹介させていただくつもりである。

## 参考文献


[1] 岩村拓哉, 廣兼賢治, 井庭崇, 竹中平蔵, 武藤佳恭. エージェントベース経済シミュレーションのためのフレームワークデザイン. 第 8 回マルチエージェントと 協調計算ワークショップ (MACC99), 1999.

[2] 井庭崇, 岩村拓哉, 廣兼賢治, 竹中平蔵, 武藤佳恭. エージェントベース社会シミュレーションのためのフレームワークデザイン. FIF working paper, フジタ未来経営研究所, 2000.

[3] Takashi Iba, Yohei Takabe, , Masaharu Hirokane, Heizo Takenaka, and Yoshiyasu Takefuji. Engineering and methodological aspects of Boxed Economy frameworks: Frameworks for agent-based economic simulations. FIF working paper, フジタ未来経営研究所, 2000.

[4] 井庭崇, 廣兼賢治, 吉川知宏, 武藤佳恭, 竹中平蔵. Boxed Economy モデルによる政策分析手法の提案. 政策分析ネットワーク 政策メッセ 99, 1999.

[5] T. Iba, M. Hirokane, Y. Takabe, H. Takenaka, and Y. Takefuji. Boxed Economy Model: Fundamental concepts and perspectives. In *Proceedings of Computational Intelligence in Economics and Finance*, 2000.

[6] 井庭崇, 中鉢欣秀, 高部陽平, 田中潤一郎, 上橋賢一, 津屋隆之介, 北野里美, 廣兼賢治. Boxed Economy の実現に向けて：エージェントベース経済シミュレーションのための基礎モデル. 電子情報通信学会「人工知能と知識処理」, 情報処理学会「知能と複雑系」合同研究会, 2001.

[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[8] Mark Grand. *Patterns in Java: a catalog of reusable design patterns, illustrated with UML, volume 1*. John Wiley & Sons, 1999.

[9] Sun Microsystems Inc. *JavaBeans* http://java.sun.com/products/javabeans/, 2000.

[10] Sun Microsystems Inc. *Java HotSpotTM Technology*. http://java.sun.com/products/hotspot/, 2000.

[11] Ivar Rumbaugh James Booch, Grady Jacobson. *Unified Modeling Language User Guide (Addison-Wesley Object Technology Series)*. Addison-Wesley Pub Co (Sd), 2000.