A Framework, Process, and Tools for Modeling and Simulating Societies as Evolutionary Complex Systems

Takashi Iba

Faculty of Policy Management, Keio University
5322 Endo, Fujisawa, Kanagawa 252-8520, Japan
iba@sfc.keio.ac.jp
http://web.sfc.keio.ac.jp/~ iba/en/

Abstract. This paper presents a framework, process, and tools for modeling and simulating societies as evolutionary complex systems. In this paper, we introduce an object-oriented computational modeling for social sciences in order to model and simulate complex systems where the model framework, "Boxed Economy Foundation Model"(BEFM), is proposed. In order to support making models based on the framework, "Model Driven Development" is proposed as a new development process. To realize the process, we also propose tools, which we call "Component Builder" (CB), that help us build the models based on the framework and process. It is the tool to generate the program code just by making the diagram and setting the parameters with a graphical user interface. Moreover, the component-based software system "Boxed Economy Simulation Platform"(BESP) is proposed for simulating and analyzing a model. The example shows that our proposed framework and tools are quite powerful and have the potential to assist thinking about complex societies.

1 Introduction

This paper presents a framework, process, and tools for modeling and simulating societies as evolutionary complex systems[1]. Although the concept of the complex systems has been highly demanded in social sciences, there is no satisfactory scheme for modeling and simulating it. Many researchers almost agree with us that the agent-based model (multi-agent model) is suitable for studying complex systems, however in the current state, there is a problem that needs to be resolved. The problem is the absence of integrated environment to support a whole research process, from conceptual modeling to simulation analysis. The problem did not become too serious up to now, because the models were of small-scale and for experimental use. It becomes, however, indispensable to resolve the problems, as the simulations come to be used practically in social science and policy analysis.

In this paper, we introduce an object-oriented computational modeling for social sciences in order to model and simulate complex systems where the model framework, "Boxed Economy Foundation Model"(BEFM), and the modeling process is proposed. In order to support making models based on the framework, "Model Driven Development" is proposed as a new development process. To realize the process, we also propose tools, which we call "Component Builder" (CB), that help us build the models based on the framework and process. Moreover, "Boxed Economy Simulation Platform"(BESP) is proposed for simulating and analyzing the models.

[1] This paper was orignally published as the paper for the conference ESSA04[7] and WEHIA04[10]

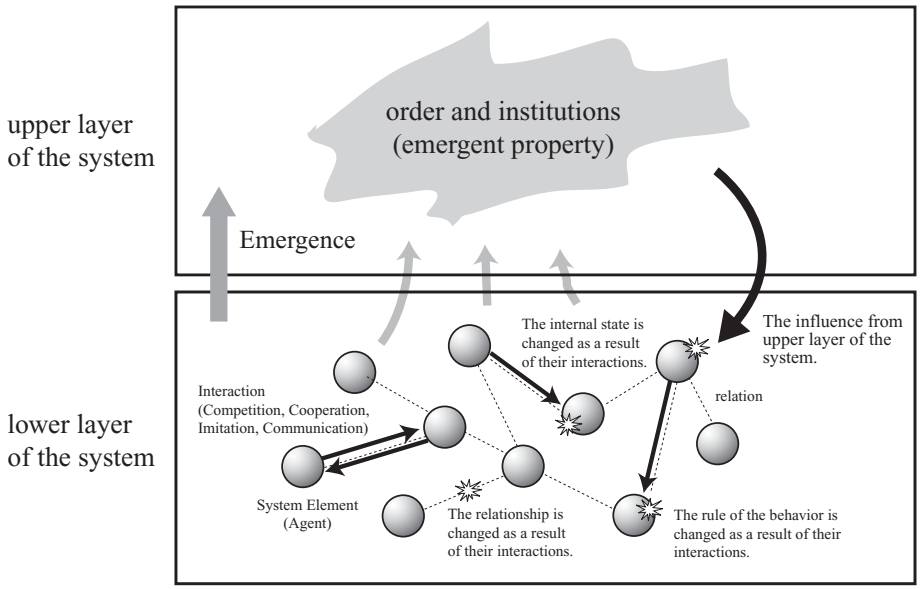

Fig. 1. An overview of complex systems

2 Our targets: evolutionary complex systems

We will begin at considering the nature of our targets. This kind of consideration is essential because "The ways of describing the problem situations (modelling languages) need to be appropriate to the nature of the problem under investigation"[16].

There is no shared definition of complex systems among the scientists, but we can say that the definition can be summarized in two ways as follows. In a broad sense, the complex system means that the system has the components where each component changes the internal states by mutually interacting with the other components. And, in a strict sense, the complex system means the system where the rules of each component's behavior are changed dynamically during the simulation. These changes often occur under the influence of the macroscopic situation that was emerged from microscopic interactions (Figure 1). Therefore the nature of the elements of complex systems are summarized in Figure 2. In this paper, we propose the framework, process, and tools for modeling and simulating the complex systems as defined above.

3 Model Framework

3.1 Model framework and its roles

We would like to propose the model framework, "Boxed Economy Foundation Model" (BEFM)[2], in order to introduce an object-oriented computational modeling for social sciences for modeling and simulating complex systems[8]. BEFM is the model framework which defines the set of concepts for modeling societies, and which supports a whole process from the analysis of target world to the execution of simulations.

[2] Our model framework is able to be applied to social phenomena in general, although it was originally developed as the framework for economic simulation[9].

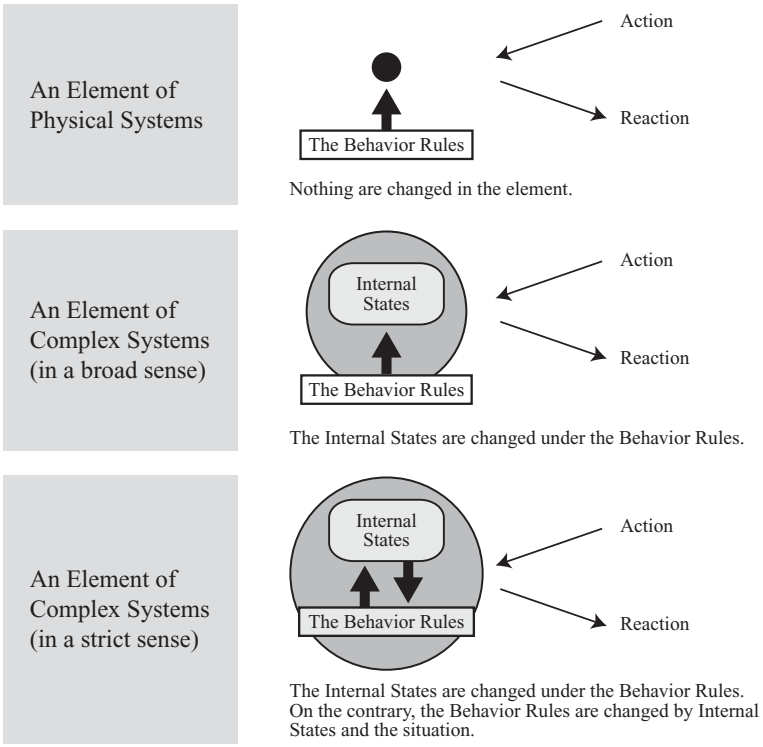

Fig. 2. System elements

In the viewpoint of conceptual modeling, BEFM provides, as a model framework, (1) a frame of reference for recognizing the target world, (2) a vocabulary for describing the concepts obtained by recognition, and (3) a code for communications among the modelers. And, in the viewpoint of simulation design, BEFM provides, as a software framework, (1) transformation rules from conceptual models to simulation models, (2) support for implementing the simulation models, and (3) architecture for sharing and reusing simulation models.

3.2 Major classes of the proposed framework

BEFM, which is an abstract of the real society from the view point of economic society, consists of 9 major elements [3](Figure 3). Their class definition and the relations between them are described as follows.

Agent "Agent" is defined to describe an autonomous actor who does an action. Each individual and social groups such as corporations, governments, families, schools, regional communities, and countries are all represented as Agents in the model.

Behavior The behavior of the agent is defined as "Behavior" in the model. Various activities such as decision-making, production, trade and communication, are described by Behavior

[3] There is another element named "Entity" in Figure 3. Entity is the abstraction of "Agent" and "Goods".

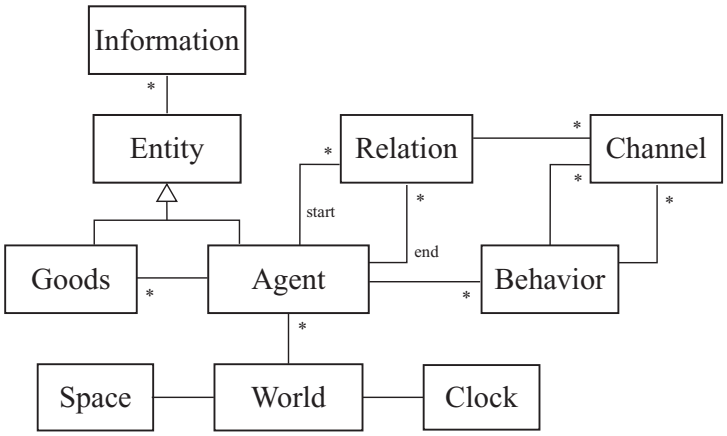

Fig. 3. Major classes of Boxed Economy Foundation Model (BEFM)

of an Agent. Different kind of Behavior can be done at the same time by the Agent. In BEFM, Behavior is defined as a state machine (Figure 5). The state machine is a system which changes the state when the event is received. The state machine changes the state by an event, which is stimulus from outside.

Goods "Goods" can be defined as material / immaterial things which are possessed by Agents in order to be used or to be exchanged with other agents. For instance, the objects modeled as Goods can be automobiles, oil, corn, financial stocks, ownership of land, books, advertisements, memorandums, water, voices, noises, garbage, money, and so on.

Information The information which is possessed by Goods or Agents is defined as "Information" in the model. Information will never exist alone, and will always be held by Goods or Agents. Examples of information possessed by Agents are "memory", "genetic information", and "name". Goods often hold Information describing various contents. For example: A newspaper can be modeled as an object which is paper (as Goods) with news articles (as Information) printed on. A conversation can be modeled as a combination of the contents (as Information) and the voice (as immaterial and transient Goods).

Relation An agent in a model usually has some kind of relationship with other agents rather than being isolated. In BEFM, the relation between agents will be described by "Relation" in the model. Relation describes relationships, for example: strangers, friends, spouses, teachers, students, employees, employers. Relation is an object by which two Agents are connected in a one-way or two-ways direction.

Channel When an Agent communicates with another Agent, "Channel" will be established between the Behavior of the Agents based on Relation. Note that Channel does not connect with Agent but connects with Behavior.

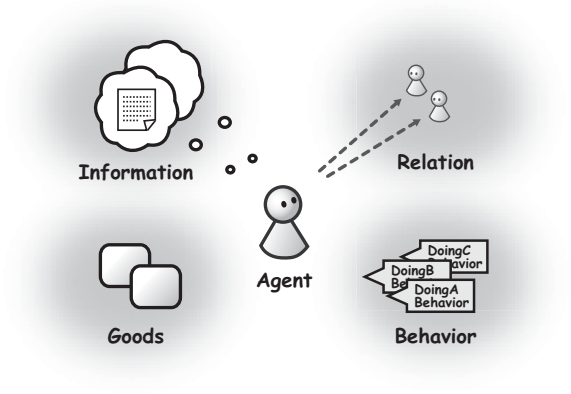

Fig. 4. The Illustration of Agent

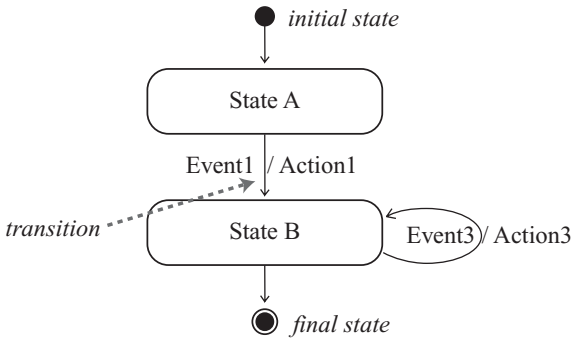

Fig. 5. Behavior as a state machine

Clock "Clock" is defined as the class to manage the flow of time in the model, during the execution of the simulation. Agent acts by the passage of the time of Clock.

Space "Space" is defined as the class to describe the spatial position in the model.

World "World" is defined as an environment in which Agents and Goods are placed.

3.3 Representing evolutionary complex systems with the proposed framework

Let us consider how we can represent an evolutionary complex system from the viewpoint of our framework. First, the complex system in a broad sense, where the reaction depends on the internal state, would be modeled as Figure 6. The agent behaves in the different way, depending on the current state of the Behavior. The agent does *Action A* when being on *State A*, and does *Action B* when being on *State B*.

Second, the complex system in a strict sense, where the rule of the behavior can be changed over time, would be modeled as Figure 7. The agent adds new behavior or exchange

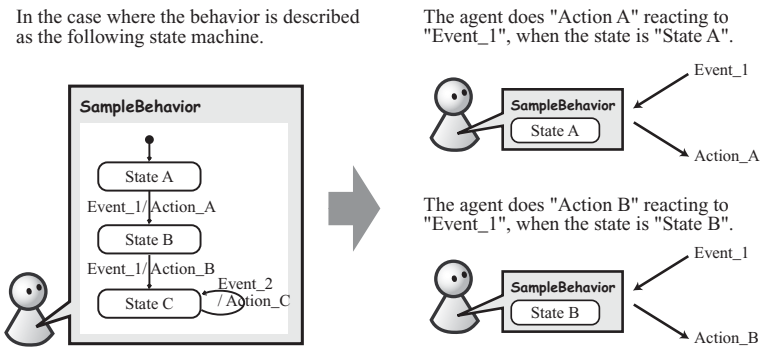

Fig. 6. Representing complex systems in a broad sense

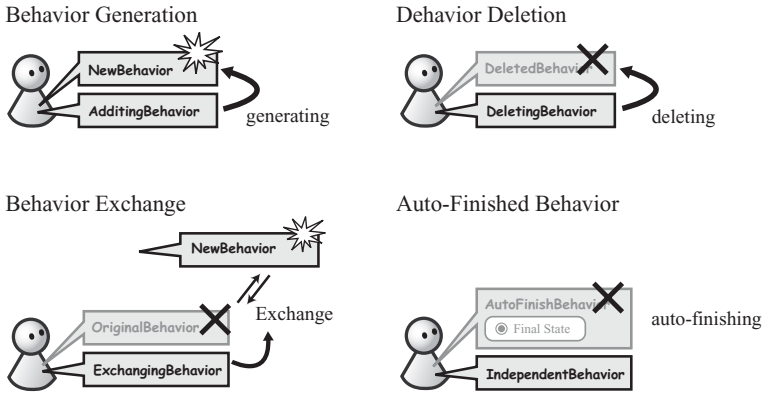

Fig. 7. Representing complex systems in a strict sense

its current behavior with other behabior, also delete the behavior. Thus the behavior of the agent can be changed.

Third, the evolution is able to be modeled as inheritance of the type of Behavior, Information, Relation, and Goods.

4 Modeling Process

Next, we would like to outline a process for modeling with the proposed framework[10]. The process consists of three major phases: “Conceptual Modeling Phase”, “Simulation Design Phase”, and “Verification Phase”. These phases are supported by our tools “Component Builder”(CB), which we will discuss in the next section.

Conceptual Modeling Phase The conceptual modeling phase is the phase to specify what the target system is. The phase consists of the class analysis, the activity analysis, and the communication-sequence analysis. In the class analysis, the modeler analyzes the target world and describes the conceptual model. He/She extracts “Agent”, “Behavior”, “Relation”, “Goods”, and “Information” from the target world, and defines them, according to the conceptual model framework of BEFM

In the activity analysis, the modeler describes the procedure of agent's behavior in an activity diagram. The activity diagram is very similar to a flowchart.

In the communication-sequence analysis, the modeler describes the sequence of the communication among the agents. The agent's behavior is often taken in cooperation with other behavior. Thus, the modeler describes the exchange of goods / information in a communication-sequence diagram.

Simulation Design Phase In the simulation design phase, the modeler translates the conceptual models into the simulation models (programs), which is executable program on the platform "Boxed Economy Simulation Platform"(BESP), which we shall explain later. The simulation design phase consists of "class design", "behavior design", and "world composition".

In the class design, the modeler describes the types and classes. It is based on the conceptual model class diagram, which has been developed in the conceptual modeling phase.

In the behavior design, the modeler describes the statechart diagram in order to describe the dynamics of the model. The modeler designs the statechart diagrams based on the following diagrams, which are developed in the conceptual modeling phase: communication-sequence diagram and activity diagram. In addition, the modeler may implement the details of the simulation as a program code [4]. Since the other parts of the simulation programs are generated by the tools, the modeler does not have to write any more codes.

In the world composition, the modeler describes the initial settings of simulated world. They are the data for building the simulation at the instance level.

Verification Phase In the verification phase, the modeler runs the simulation and inspects whether the simulation program is coded rightly. If necessary, the modeler returns to the first or second phase and modifies the models.

5 Modeling Tools

In order to support making the simulation models with the proposed framework and process, we would like to propose "Component Builder"(CB) [10]. For conceptual modeling, CB provides the drawing tools in UML (Unified Modeling Language). In addition, for simulation design, CB provides the setup by which the programming to make the simulation is greatly reduced. As a result, the modeler will be able to make the simulation as long as they have the basic skills of programming, and they do not have to make the design or the implementation concerning the structure which makes the programming more difficult. Moreover, the user can make and change their simulation promptly, and then can give priority to the analysis of the consequences.

CB consists of four designers and a composer: "Model Designer", "Behavior Designer", "Activity Designer", "Communication Designer" and "World Composer"(Figure 8). They are the tools to generate the program code just by making the diagram and setting the parameters with a graphical user interface.

[4] Current tools, which we propose in this paper, are not sophisticated enough to transform from design models to program code for a hundred percent. The modeler should write the program code of the action description of agent's behavior. We, however, now try to develop more sophisticated tools which help modelers to build simulations without any programming.

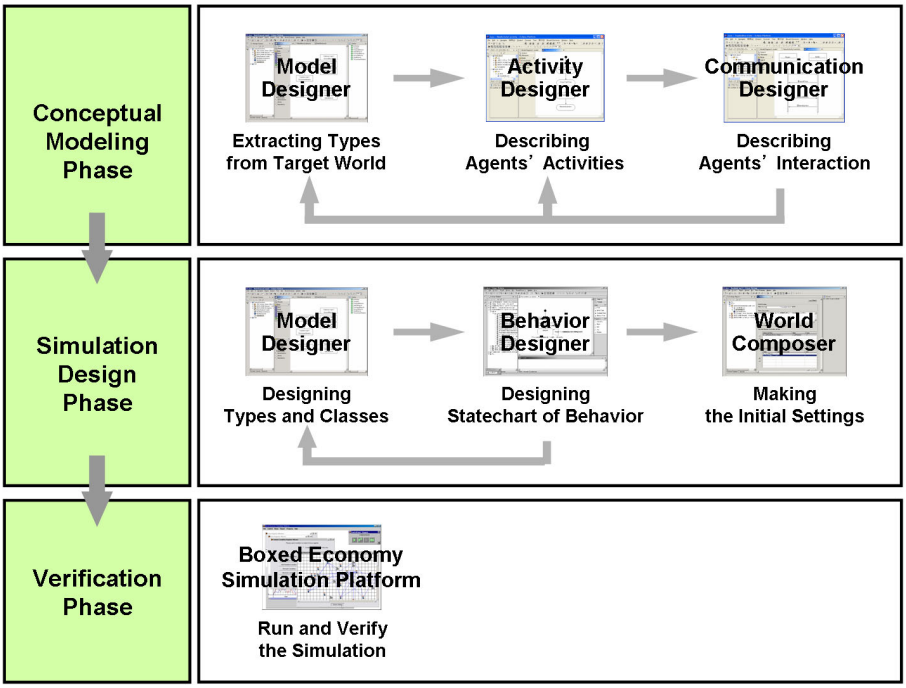

Fig. 8. Modeling process and tools

5.1 Model Designer

Model Designer is a tool for modeling the static view of the simulation. With using Model Designer, the modeler can generate the program code automatically just by drawing the class diagram and setting the model with a graphical user interface.

5.2 Activity Designer

Activity Designer is a tool for modeling the activity of Agents. For conceptual modeling, modelers can use this tool to draw the activity diagram based on UML.

5.3 Communication Designer

Communication Designer is a tool for modeling the interaction among Agents. For conceptual modeling, modelers can use this tool to draw the sequence diagram based on UML.

5.4 Behavior Designer

Behavior Designer is a tool for modeling the dynamic view of the simulation. With using Behavior Designer, the modeler can generate the program code automatically just by drawing the state chart diagram and setting the model with a graphical user interface.

5.5 World Composer

World Composer is a tool for modeling the initial state of the simulation world. With using World Composer, the modeler can generate the program code automatically just by setting the parameters with a graphical user interface.

6 Simulation Platform

In order to execute the simulations based on the proposed framework, we would like to propose "Boxed Economy Simulation Platform" (BESP) as a sharable basis for agent-based social simulation[11]. BESP is a software platform to make, to execute, and to analyze the agent-based social simulations.

BESP is designed to realize an extensible software application with component-based architecture. The user can obtain the simulation environment which suits the needs, only if he/she sets necessary components into the platform. There are two kinds of components built into the platform: they are "model component" and "presentation component". The model component is a software component that implements the model which the user wants to simulate. The model component is made based on BEFM. The presentation component is a software component for the user interface to operate and to visualize the simulation, and to output the results into the file.

Model components and presentation components are independent of each other, communicating indirectly by sending and receiving the events through BESP. Therefore, the user simulates his/her original social model with existing presentation components even if he/she makes only the model components. In contrast, the user makes his/her original user interface as presentation components that do not specialize in a specific social model.

7 An example: Evolutionary simulation of strategy

Now we would like to demonstrate how we can model a society as an evolutionary complex system with the proposed framework and tools. The example is evolutionary simulation of strategy in the Iterated prisoner's dilemma[6].

7.1 Target world

In the iterated prisoner's dilemma, the players make their choice of "Cooperation"(C) or "Defection"(D). The choice of defection yields a higher points than cooperation. If both defect, however, both get lower points than if both had cooperated. The iterated prisoner's dilemma is said to be "an elegant embodiment of the problem of achieving mutual cooperation, and therefore provides the basis for the analysis."[3]

The player makes the choice based on its own "strategy", that is the rule of the behavior. The players are paired with each other in a round robin tournament. A certain number of the matches are held in each game and then the score is updated and recorded. This is a typical version of the model[2].

In this paper, we extend the model in which players will improve their strategies after the tournament. There are two scenarios about the strategy improvement. The first is "Imitation based on match result", and the second is "Imitation based on the tournament result". In the experiment, the consequences of two scenarios are compared.

7.2 Model Overview

First, we describe the target world with the types "Agent" "Behavior" "Goods" "Information" "Relation" according to BEFM. About "Agent", the player of the game is modeled as *PlayerAgent*(Figure 9: Left). In addition, the actor who manages the tournament and games is modeled as *RefereeAgent*.

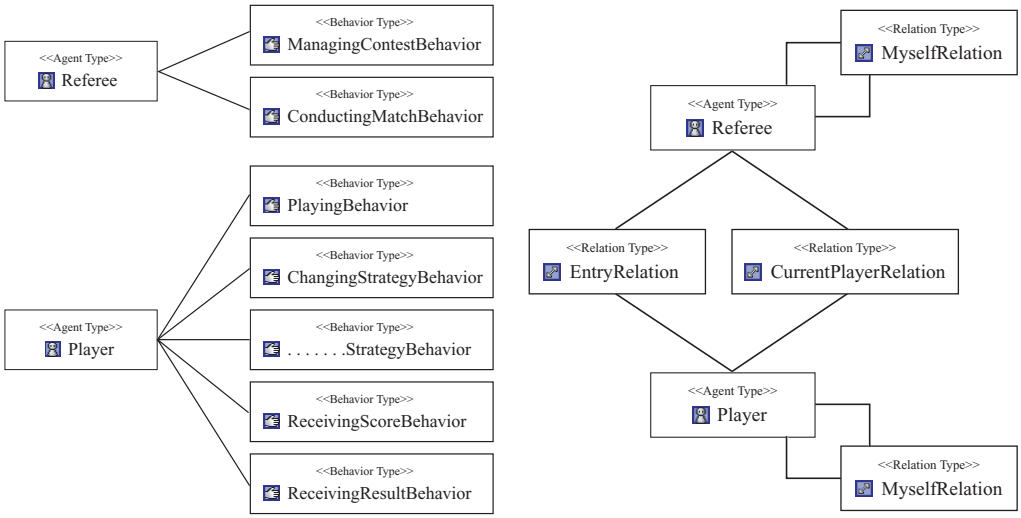

Fig. 9. Class diagram of the extended IPD model

The "Behavior" of each agent are represented as follows. *PlayerAgent* has some kind of behavior: *PlayingBehavior* for playing the games, *StrategyBehavior* for deciding the choice on the next move, and *ChangingStrategyBehavior* for updating the strategy. The strategy is embodied as the automaton in the *StrategyBehavior* (Figure 10). *RefereeAgent* has two kind of behavior: *ManagingContestBehavior* for managing the tournaments and *ConductingMatchBehavior* for making players play.

We next describe the flow of communication between agents in order to specify "Goods" and "Information" which are passed between agents (Figure 11: Right). The figure shows that there are no goods in this model. We can find some type of information: *StartingSignal*, *Choice*, *Preceding Choice*, and *Score Report*.

At the final step, we describe the relation between agents as "Relation" (Figure 10: Right)。There are following three relations: *EntryRelation* from *RefereeAgent* to *PlayerAgent*, *CurrentPlayerRelation* from *RefereeAgent* to the *PlayerAgents* who are currently playing the games, and *MyselfRelation* for the both of *RefereeAgent* and *PlayerAgent*.

Now we check the model from the viewpoint of evolutionary complex systems. The choice of *PlayerAgent* is based on the strategy and the history of the game so far. It means that the reaction is based on the internal state, so this model includes the characteristics of "complex system" in a broad sense. Moreover, the *PlayerAgent* changes the own strategy in this model. It means that the rule of the behavior can be changed, so this model is also includes the characteristics of "complex system" in a strict sense. And, the model has evolutionary aspects, because the strategy is inherited by the imitation.

7.3 Simulation Flow

Let us explain the flow of simulation, which is described in Figure 11. At first, when *RefereeAgent* receives *TimeEvent*, the *ManagingContestBehavior* makes the pair enter into the round-robin matches. Then, *ConductingMatchBehavior* receives the information of a pair and connect *CurrentPlayerRelation* from the agent to the *PlayerAgents* of the pair.

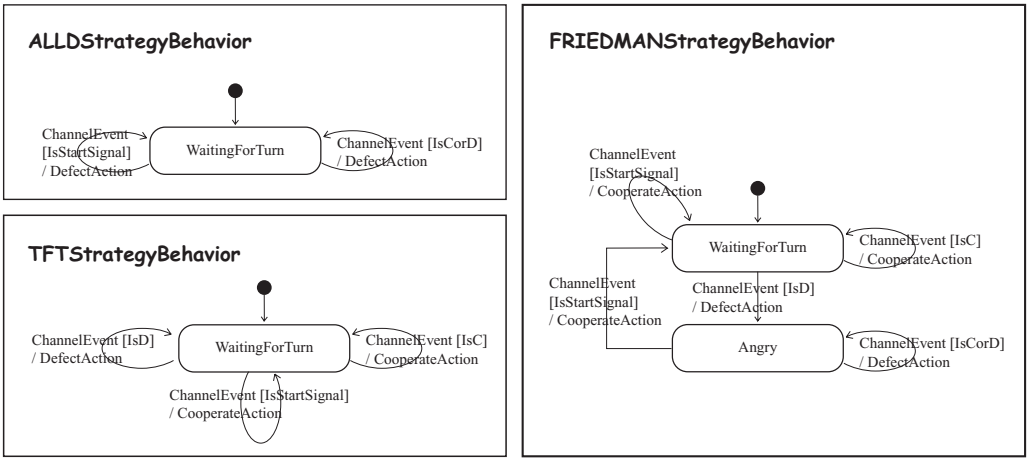

Fig. 10. Statechart diagrams of Strategy Behavior

RefereeAgent asks *PlayerAgent* about the choice of the next move. *PlayingBehavior* of *PlayerAgent* receives the demand, and asks the own *StrategyBehavior* about the choice of his/her next move. Then *StrategyBehavior* replies the choice, which is based on its own state, through *PlayingBehavior*. *ConductingMatchBehavior* of the *RefereeAgent* checks the choices and calculates the score based on the outcome of this move.

After the first turn, *RefereeAgent* asks about the choice of the next move with sending the information of the preceding choices. *PlayerAgent* replies his/her choice to *RefereeAgent* in the same way as at the first turn. Then, *RefereeAgent* updates the score. The process above is iterated during certain moves.

Next, *RefereeAgent* sends the information about the scores to the *PlayerAgent*. The *PlayerAgents* receives and records it. Then, *RefereeAgent* reports the scores to the own *ManagingContestBehavior*. *ManagingContestBehavior* updates the list of total score of each *PlayerAgents*. Thus the match of two players is finished. This process is iterated in a round-robin tournament.

After the round-robin tournament, *RefereeAgent* sends the list of total scores in the tournament to all *PlayerAgents*. *PlayerAgent* receives and records it. After the tournament process, *PlayerAgents* receives *TimeEvent*. *PlayerAgent* imitates the other's strategy.

7.4 Simulation Result

We would like to explain the simulation result only summarily, because the result itself is not important in this paper. The simulation settings have two players per strategy, where the 9 types of strategies are provided as follows: ALL-C, ALL-D, RANDOM, TFT, TF2T, FRIEDMAN, JOSS, PER-CD, PER-CCD. As we have mentioned before, two types of settings are experimented: (1) "Imitation based on match result" and (2) "Imitation based on the tournament result". In the scenario (1), all agents will adopt a strategy "ALL-D" after some steps. Therefore the average score will be much lower than the one at the initial step. In the scenario (2), on the contrary, agents will adopt a strategy "FRIEDMAN" or "TFT" (Figure 12). Therefore the average score will be higher than the one at the initial step. Thus we see that sharing the macroscopic information is effective to achieve mutual cooperation.

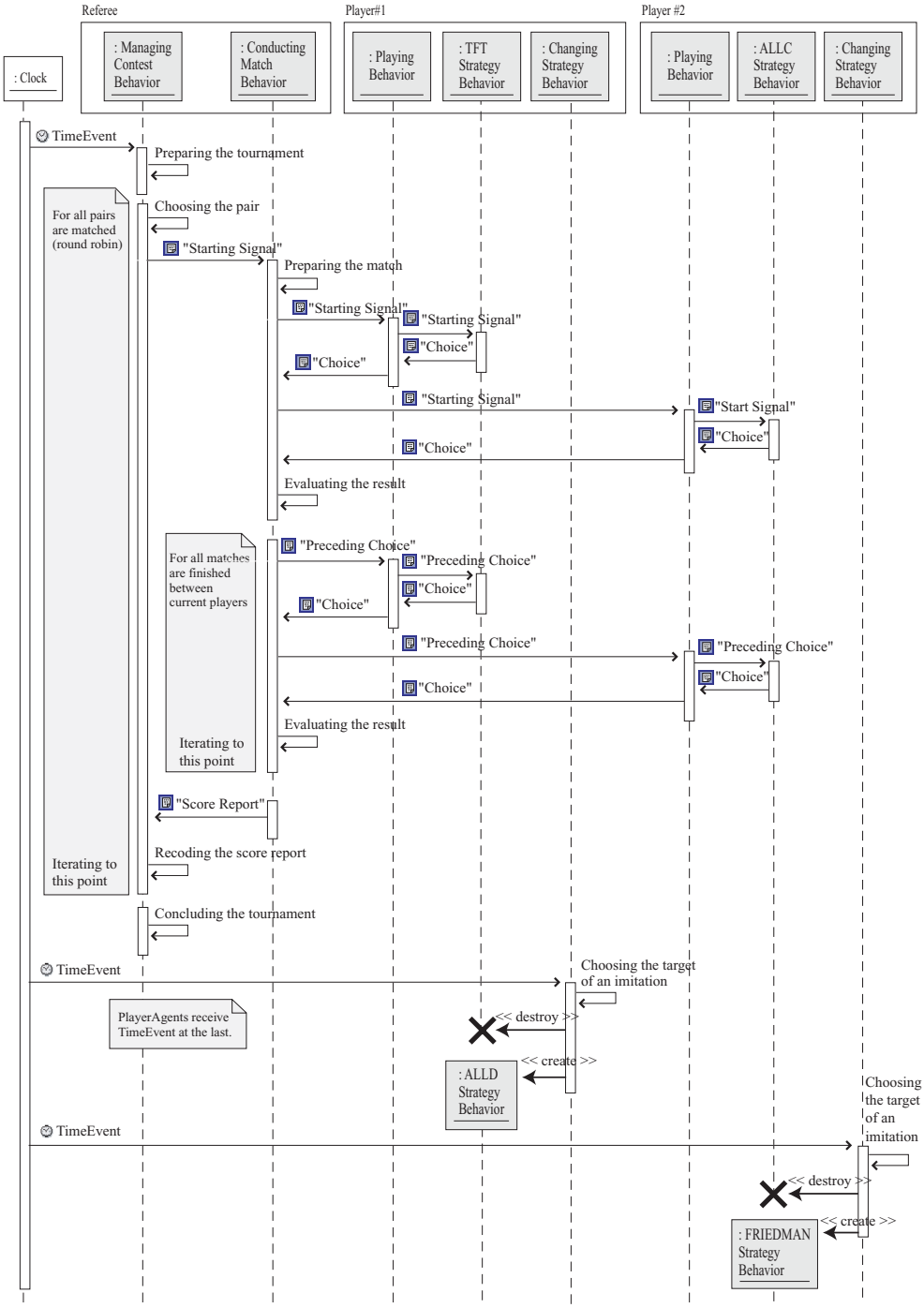

Fig. 11. Communication-sequence diagram of the extended IPD model

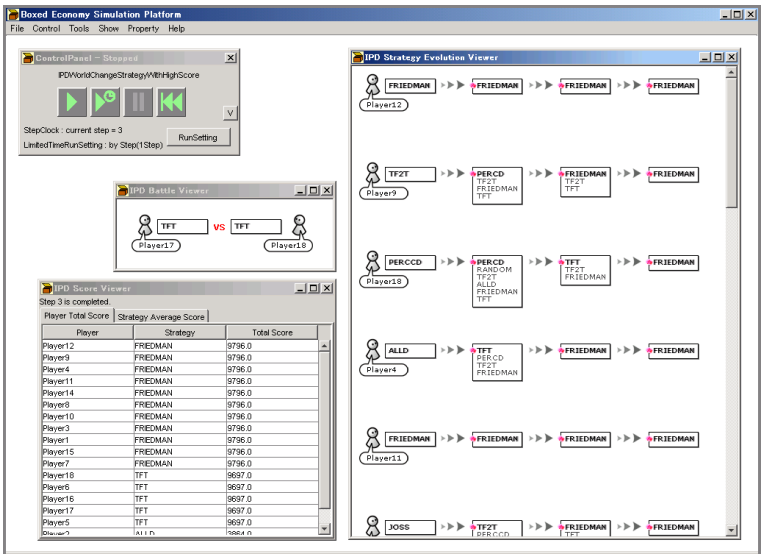

Fig. 12. Screenshot of the extended IPD simulation on BESP

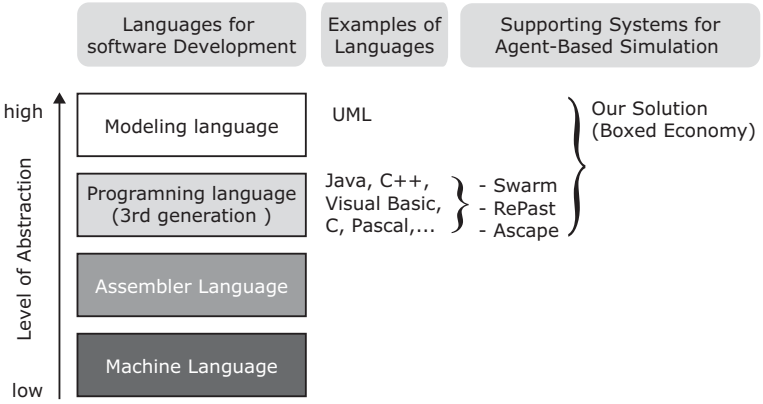

Fig. 13. Level of Abstraction, Languages, and Supporting Systems

8 Comparison with the existing systems

In the final place, we shall make clear how our framework and tools are different from the existing systems. In the last some years, several languages, frameworks and tools for agent-based simulations have been proposed. For instance, "Swarm Simulation System" provides the class library for the simulation of complex adaptive systems[13]. As well as Swarm, "RePast" provides the class library to make the agent-based model[4]. "Ascape" provides the framework, and it is said that the amount of the code description can be less than that of Swarm and RePast[15]. "LSD" is a language for developing simulation models[1].

These supporting systems assist the modelers to write programs by providing a general library and framework, and in fact these systems are useful for the reduction of programming. These systems, however, would not support the modelers to do modeling. On the other hand,

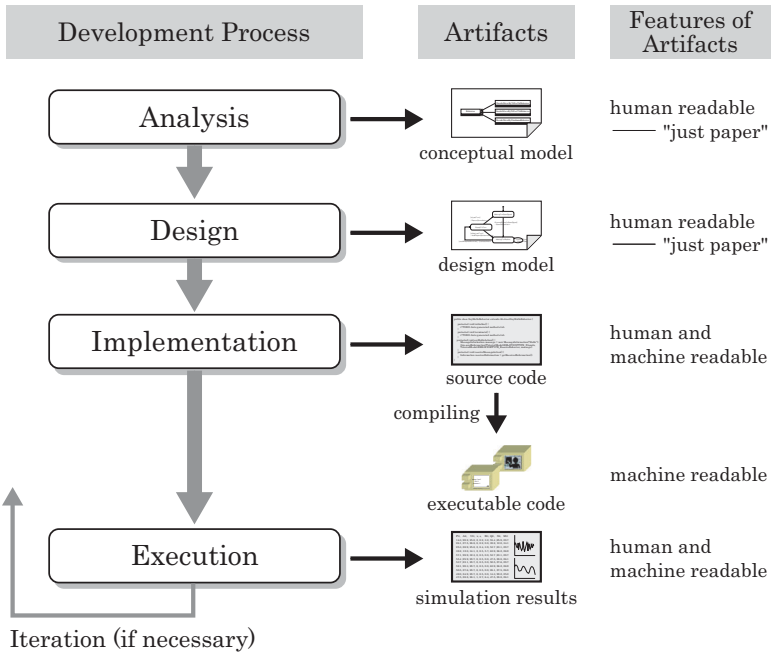

Fig. 14. Traditional Process of Software Development

our solution, i.e. providing a model framework, process, and tools, is to support a whole process from conceptual modeling to simulation implementation (Figure 13).

From the viewpoint of software engineering, the existing systems are based on the traditional development process, which is driven by implementation — program coding (Figure 14). In the traditional development process, the transformation from design model to program code is done by hand. According to this process, the modeler should write a lot of program codes.

On the contrary, our solution is based on the emerging development process, which is driven by modeling: Model Driven Development[10] [5] . We can use more high-level language for development, instead of writing in the lower-level language, i.e. program code at the moment (Figure 15). In the new process, we can concentrate on modeling without considering the software implementation, because the program code will be an exact translation of the model. In this sense, design models are development artifacts which is the same as design artifacts, therefore the design model is no longer "just paper" or "blueprint". Therefore, the design models are development artifacts that contribute directly to simulation development.

[5] In the field of software engineering, there is a trend toward considering the design models as the development artifacts that contribute directly to software development. "MDA" (Model Driven Architecture) and "Executable UML" (Unified Modeling Language) are proposed for the Model Driven Development [5, 14, 12]. The point is "using modeling languages as programming languages rather than merely as design languages."[5]. As a result, "It makes it possible to raise the level of abstraction for software development"[5] (Figure 13). History tells that the productivity and quality are improved in consequence of raising the level of abstraction.

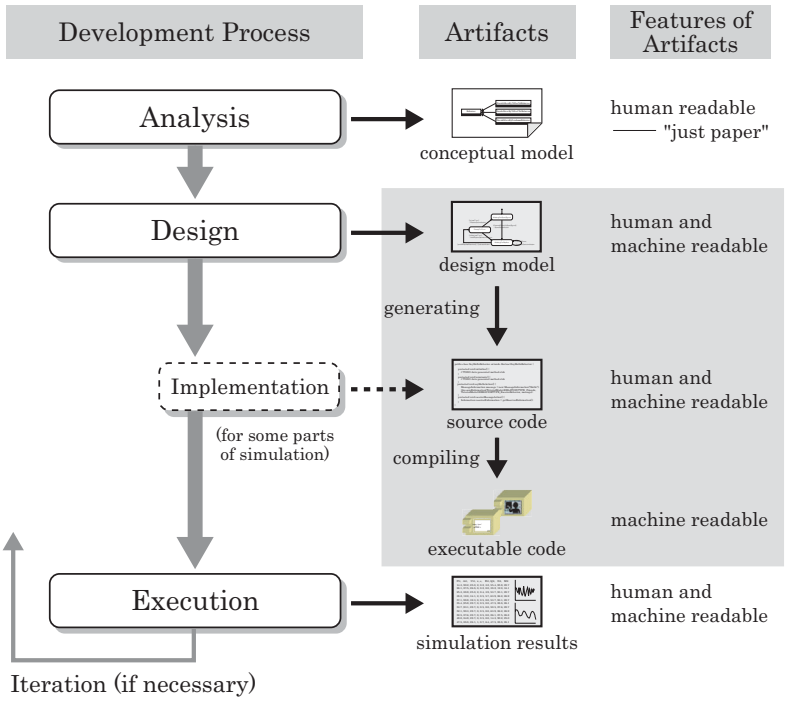

Fig. 15. New Process of Software Development

9 Conclusion

In this paper, we proposed a framework and tools for modeling and simulating societies as evolutionary complex systems. We have already applied the proposed framework and tools to several social phenomena as follows: the electric power market model, the model of the video cassette format competition, the evolutionary market model (Nelson-Winter model), the evolving network model, the evolutionary model of iterated prisoner's dilemma, the model of emergence and collapse of money from barters, the queuing model of airport, and the artificial stock market model. And, we have introduced our framework and tools into my classroom at the university. Our experiences show that our proposed framework and tools are quite powerful and have the potential to assist thinking and communicating about complex societies and economies.

The frameworks and tools are opened to public. Creating the foundation for studying complex systems is an oversized project for our members to complete. We would like to realize this by collaborating with many researchers in various fields. Please contact us on http://www.boxed-economy.org/, if you are interested in our challenge.

References

1. E. S. Andersen and M. Valente. The art of simulation and the lsd system, 2003. http://www.business.aau.dk/evolution/.
2. Robert Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.
3. Robert M. Axelrod. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton University Press, 1997.
4. N. Collier. Repast: An extensible framework for agent simulation. *The University of Chicago's Social Science Research*, 2003. http://repast.sourceforge.net/.
5. David S. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. Wiley Publishing, 2003.
6. T. Iba. A study on simulating economies and societies as evolutionary complex systems. *Graduate School of Media and Governance, Keio University*, 2003. in japanese.
7. T. Iba. A framework and tools for modeling and simulating societies as evolutionary complex systems. In *2nd. International Conference of the European Social Simulation Association*, 2004.
8. T. Iba, Y. Chubachi, Y. Takabe, K. Kaiho, and Y. Takefuji. Boxed Economy Foundation Model. In *The AAAI-02 Workshop on Multi-Agent Modeling and Simulation of Economic Systems*, pages 78–83, 2002.
9. 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*, pages 941–944, 2000.
10. T. Iba, Y. Matsuzawa, and N. Aoyama. From conceptual models to simulation models: Model driven development of agent-based simulations. In *9th Workshop on Economics and Heterogeneous Interacting Agents*, 2004.
11. T. Iba and Y. Takefuji. Boxed Economy Simulation Platform for agent-based economic and social modeling. In *Computational Analysis of Social and Organizational Systems 2002*, 2002.
12. Anneke Kleppe, Jos Warmer, and Wim Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley, 2003.
13. F. Luna and B. Stefasson, editors. *Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming*. Kluwer Academic Publishers, 2000.
14. Stephen J. Mellor and Marc J. Balcer. *Executable UML: A Foundation for Model-Driven Architecture*. Addison-Wesley, 2002.
15. M. T. Parker. What is ascape and why should you care? *Journal of Artificial Societies and Social Simulation*, 4(1), 2001. http://www.soc.surrey.ac. uk/ JASSS/4/1/5.html.
16. B. Wilson. *Systems: Concepts, Methodologies, and Applications*. John Wiley & Sons, 2 edition, 1990.