Evolutionary Economics and Social Complexity Science 19

Stanislaw Raczynski

Interacting Complexities of Herds and Social Organizations

Agent Based Modeling

Springer

Evolutionary Economics and Social Complexity Science

Volume 19

Stanislaw Raczynski

Interacting Complexities of Herds and Social Organizations

Agent Based Modeling

Springer

Stanislaw Raczynski
Facultad de Ingeniería
Universidad Panamericana
Ciudad de México, México

ISSN 2198-4204 ISSN 2198-4212 (electronic)
Evolutionary Economics and Social Complexity Science
ISBN 978-981-13-9336-5 ISBN 978-981-13-9337-2 (eBook)
https://doi.org/10.1007/978-981-13-9337-2

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

According to John von Neumann, “by a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work — that is, correctly to describe phenomena from a reasonably wide area.” Humans always (sometimes unconsciously) have used models created in their brains. When our technical skills have grown, the models acquired the form of physical, scale models, drawings, and finally sophisticated logical and mathematical constructions. The common concept of modeling is defined as a scientific activity, the aim of which is to make a particular part or feature of the world easier to understand.

The complexity of the real world can be modeled to some extent. There are many definitions of complexity, recently related to “system of systems” structures. Note that a system that contains a great number of sub-systems or items or a huge number of differential equations is not necessarily complex. The complexity lies in the way the components interact with each other and the diversity of system components. In such systems, the simulation results may provide information about the behavior of the whole system, which is not the sum of individual behavior patterns. This is also interpreted as nonlinearity. This book is focused on this kind of modeling and simulation experiments.

Analog and digital computers gave us a powerful tool for model building and analysis. At the very beginning of the computer era, the differential equations have been solved on analog machines, helping scientists and engineers to design mechanisms, circuits, and complex devices. The field of model applications has grown over the decades, including not only the works of engineering and exact sciences but also the models of animal and human societies.

At the very beginning, model builders have been looking for some kinds of algebraic, ordinary, or partial differential equations to describe real system behavior. The most known and explored field is the System Dynamics (SD) approach that mainly uses models in the form of ordinary differential equations. However, it should be noted that this is not the only way to build models. A strange conviction aroused among the modelers that everything in the real world can be described by differential equations. In general, this is not true. Although the SD methodology is still widely used and useful, there are other ways for model building, like fuzzy logic, differential inclusions, discrete event simulation, and agent based models, among others.

The topic of this book is agent based modeling. The rapid growth of the computational capacity of new computers permits us to create thousands of objects in computer memory and make them interact with each other. In agent based models, the objects are equipped with certain artificial intelligence, can optimize their behavior, and take decisions. Some systems can be modeled both using differential equations and agent based approach. The results of these two methods are frequently quite different, for example, results of the Lotka-Volterra prey-predator model and the prey-predator agent based model. Here, we will not suggest which of these models is valid or not. These are just different modeling methods that produce results of different kind. Undoubtedly, agent based modeling is more flexible and can reflect more behavioral patterns of the individuals, providing the insight on the macro-behavior of the system. In Chap. 1, there are comments on some agent based modeling tools. The other chapters contain examples of applications to artificial societies and competing populations of individuals and the growth, interactions, and decay of organizations and other applications. For reader's convenience, a short recall about object- and agent-based modeling is repeated in each chapter. Thus, each chapter can be read as independent unit. In Chap. 9, you can find a description of an experimental software package that uses the classic continuous system dynamics graphical user interface (GUI) that is used to construct the model. However, the transparent simulation engine that runs behind this GUI is discrete event simulation. This way, we can compare the results of the conventional system dynamics packages with these provided by discrete event simulation. The relevant differences between these two simulation paradigms are pointed out.

Mexico City, Mexico Stanislaw Raczynski

Acknowledgements

I would like to express my gratitude to the Editors of the journals listed below for the permission to use the updated versions of my articles, as follows:

Simulating self-organization and interference between certain hierarchical structures. Nonlinear Dynamics, Psychology, and Life Sciences, 2014, Vol 18, no 4, used in Chap. 2 of this book,

A Self-destruction game, Nonlinear Dynamics, Psychology, and Life Sciences, 2006, Vol 10, no 4, used in Chap. 7 of this book,

The spontaneous rise of the herd instinct: agent-based simulation, Nonlinear Dynamics, Psychology, and Life Sciences, to appear, used in Chap. 5 of this book.

Simulation of the dynamic interactions between terror and anti-terror organizational structures, Journal of Artificial Societies and Social Simulation, Vol. 7, no. 2, used in Chap. 3 of this book.

Influence of the gregarious instinct and individuals' behavior patterns on macro migrations: simulation experiments, Journal of Human Behavior in the Social Environment, Vol. 28, no 2, used in Chap. 6 of this book. Visit the journal home page at www.tandfonline.com.

Stanislaw Raczynski

Contents

Chapter 1
Agent-Based Models: Tools

1.1 General Remarks

The methodological focus of this book is the object- and agent-based simulation. No state equations or system dynamics schemes are used. Recall that in the discrete object-based modeling, we create objects that behave according to the user-defined rules and execute their events in discrete moments of the model time. The agent-based models manage objects called agents, which are equipped with certain "intelligence." They can take decisions, optimize their actions, and interact with each other and with the environment. Agent-based models (ABMs) are a type of microscale models that simulate the simultaneous operations and interactions of multiple agents in an attempt to recreate and predict the appearance of global complex phenomena.

The individuals in ABM models may be of different types. Although the rules of behavior are the same for individuals of the same type, the behavior is not identical for all of them. This modeling method has many applications, mainly in ecology, biology, and social sciences. A key notion is that simple behavioral rules (micro model) generate complex (macro) behavior. An important central tenet is that the whole is greater than the sum of the parts. Individual agents are typically characterized as rational. They are presumed to be acting in what they perceive as their own interests, such as reproduction, economic benefit, or social status, using heuristics or simple decision-making rules (Railsback et al. 2006; Bandini et al. 2009). Note the main difference between object-oriented and simulation package. The latter, in addition to object creation, provides (or should provide) a "clock" mechanism that automatically manages the model time and event execution. The ABM modeling is supported by many programming and simulation tools. Let us list only some of the most popular tools: SWARM developed in 1994 by the Santa Fe Institute (Swarm Development Group, 2001), Ascape developed in 2001 (Parker 2001), Breve-2.7.2 (Klein 2002), Recursive Porous Agent Simulation Toolkit released in

S. Raczynski, *Interacting Complexities of Herds and Social Organizations*,
Evolutionary Economics and Social Complexity Science 19,
https://doi.org/10.1007/978-981-13-9337-2_1

2003 (Michael et al. 2006), Cormas developed in 2004 by VisualWorks (Bommel et al. 2015), MASON (Luke et al. 2005), MASS package (Tatai et al. 2005), FLAME (Coakley et al. 2006; Holcombe et al. 2013), MATSim of EHT Zürich (Bazzan and Klugl 2009), and SOARS developed in 2010 (Tanuma et al. 2005, 2006), among others.

ABMs are widely used in modeling of the organization dynamics. An example of an agent-oriented model, called the BC model, can be found in the article by Krause (2000). In that model, the agent's attributes include "opinions," and the interaction between agents depends on the distance between their opinions in a nonlinear way. These interactions can result in an action being taken by the agent. Other examples of models of social structures based on the concept of opinion interactions can be found in Latane and Nowak (1997) and Galam and Wonczak (2000). A similar approach is presented by Chatterjee and Seneta (1977) and Cohen et al. (1986). These works refer to the dynamics of forming of social groups in accordance with the existing agents' attributes (opinions). Some quite interesting results, more closely related to the terrorism problem, are described by Deffuant et al. (2002).

Some more general concepts of "computational sociology" and agent-based modeling (ABM) can be found in the article of Macy and Willer (2002). Other general recommended readings in the field are Bak (1997), Cioffi-Revilla (1998), Gotts et al. (2003), Axelrod (1997), Epstein and Axtell (1996), and Holland (1998). An interesting contribution to a model of the structure of the Osama bin Laden organization is included in a Vitech Corporation page (link: see Long 2002). Other (ABM)-oriented approach can be found in Crowder et al. (2012) and Hughes et al. (2012). In these publications we can find discussions about the potential advantages of the ABM approach through a range of examples and through the identification of opportunities in the field of organizational psychology.

Another approach is used by Lustick (2000), where the agents interact on a landscape. It is shown that macro-patterns emerge from micro-interactions between agents. An important conclusion is that such effects are more likely when a small number of exclusivist individuals are present in the population. The simulations of other mechanisms of clustering in agent-oriented models are described by Younger (2003), who deals with the creation of social structures in the process of food and material storage.

1.2 Discrete Event Simulation

Recall that by the *model time*, we understand the time variable that is controlled by the simulation program during the simulation run. The *real time* represents the time of our (or computer) physical clock. For example, simulating the movement of a galaxy, we can simulate several millions of model time years. On a fast computer, his simulation may take several minutes in the real time.

There are many real systems, where we can define the processes named *events* that consist in changing the state of the system. For example, the events may describe the start or the end of a service process and a birth or death of a model entity or taking place in a waiting line. In many situations such events can be considered to be executed in a very small interval of time, compared to the total length of model simulation time. The discrete event simulation means that we suppose that the model events are discrete, i.e., they are accomplished within model time interval of length zero. This model simplification makes the simulations very fast.

The Discrete Event Specification (DEVS) formalism is used to describe models in discrete event simulation. In the DEVS formalism, an "atomic" model M is defined as follows:

$$\begin{gathered} M = \langle X, S, Y, \sigma_{\text{int}}, \sigma_{\text{ext}}, \lambda, \tau \rangle \\ \sigma_{\text{int}} : S \to S, \sigma_{\text{ext}} : Q \times S \to S, \quad \lambda : Q \to Y, \end{gathered} \tag{1.1}$$

where X is the input space, S is the system state space, Y is the output space, σ_{int} is the internal state transition function, σ_{ext} is the external transition function, and Q is the "total state."

Atomic models can be coupled to form a coupled model. The coupled models can also be coupled in hierarchical way to form more complex models. The coupled DEVS model is as follows:

$$coupled\,DEVS \equiv \left\langle X_{self}, Y_{self}, D, \{M_i\}, \{I_i\}, \{Z_i j\}, select \right\rangle$$

The subindex self-denotes the coupled model itself. D is a set of unique component references. The set of components is:

$$\{M_i \mid i \in D\}$$

The select component defines the order of execution for simultaneous events that may occur in the coupled model. This component must be added to the model to avoid ambiguities in the simulation algorithm and to make the model implementation-independent. There is a huge research done on the select algorithms because the treating of the simultaneous events is rather difficult task.

To treat complex models with variable structure, the Dynamic Structure Discrete Event System Specification (DSDEVS) is used. We will not discuss the DSDEVS formalism here. The use of the DEVS formalism is relevant in big models, where the time of execution, hierarchical model building, and portability are important factors.

By *time and event management* (TEM), we understand the time clock and event queue management (inside the "simulation engine"), including the basic queuing model operations provided by the simulation package. The object behavior modeling (OBM) is a set of additional items like user-defined distributions and logical functions, nontypical operations, object attributes, and the general object behavior.

Let us start with GPSS (General Purpose Simulation System), omitting earlier tools like the forgotten but very nice language of the 1950s CLS (control and simulation language).

1.2.1 GPSS

This language, developed primarily by Geoffrey Gordon at IBM around 1960 (Gordon 1975), has contributed important concepts to every discrete event simulation language developed ever since. This is an old tool, but it is still used and works perfectly. In fact, GPSS is an object-oriented tool, although it does not fit into the modern object-oriented paradigms. The objects in GPSS are called transactions. These are moving items that appear, go through the fixed model facilities, and disappear. GPSS World has been extended by PLUS, the Programming Language Under Simulation. The TEM level instruction set of GPSS is simple and easy to use. It can be dominated by anyone in few hours of learning and running example queuing models. The OBM level mechanisms are not so easy. Recall that the new versions of GPSS have an embedded language PLUS. If the user wants to equip objects (transactions) with any additional properties and individual, nonstandard behavior, he must learn PLUS and dominate the information about the SNAs (standard numeric attributes). The PLUS manual is a whole chapter of the GPSS manual or a separate document of about 60 pages. The SNA documentation occupies also several dozen pages, including great number of attributes and additional items. Using all this stuff, the user can simulate more advanced models, but the created objects can hardly be considered as "intelligent."

1.2.2 Arena

Arena modeling system from Systems Modeling Corporation is a nice and widely used simulation tool. It is equipped with a graphical user interface (GUI) and animation mechanisms (see Kelton et al., 2004). The TEM level of Arena permits to quickly create a queuing or manufacturing discrete event models, needs no coding, and results in clear flowcharts of the model. The OBM level is somewhat more complicated. Arena is built on the SIMAN (Pedgen et al. 1995) simulation language. So, first of all, the user must learn SIMAN to be able to manage user-defined logics, statistics, and/or a nonstandard object behavior. The Arena entities (moving objects) can be equipped with time attributes, cost attributes, entity-type variable, group member variables, and other. The specification of the attributes and other Arena pre-defined variables takes about 30 pages in the Arena documentation. Again, if the user wants to create and manage a little bit more complicated object behavior, he/she must learn SIMAN and dominate dozens of pages of the Arena manual.

1.2.3 SIMIO

This is a multi-paradigm software delivered by SIMIO LLC. SIMIO® is created by a team of simulation software developers led by Dennis Pedgen and Sturrok (2010).

Compared to Arena, SIMIO is a step forward in creating models with intelligent objects. The object definition in SIMIO is more general. Objects may be fixed facilities or moving dynamic objects named entities. The user can define his/her own objects, store and reuse them, or use the objects from the standard library. These may be fixed (server, machine), link (a pathway for entities), node (link intersections), entity (dynamic object, like client in a shop), or transporter (it can pick up and drop entities at nodes).

The user defines the object properties. They may be of different types such as strings, numbers, selections from a list, and expressions. The properties are edited in multiple edition windows. There are many ways to define a SIMIO model. A programmer familiar with an object-oriented language like C++ or Delphi can understand and dominate the SIMIO modeling in reasonable time and effort. SIMIO creators claim that the process-based objects in SIMIO are both simpler and more powerful than the code-based objects in other modeling tools. SIMIO offers both TEM and OBM facilities, although they are not clearly separated from each other.

1.2.4 Simula

We must mention here Simula, its mostly known version 67 (Dahl and Nygaard 1967). Although it is a tool developed more than 50 years ago, it is still perhaps one of the most advanced and elegant object-oriented languages. In fact, Simula itself is just object-oriented and not a simulation language. The modeling facilities have been added to the language as a part of its standard class library and are encapsulated in the Process class. Any object that inherits the Process class properties can use the clock mechanism and event scheduling. The object behavior management is coded directly in the language. As for an old software, it originally had no GUI and other graphical facilities. The language is rather difficult to learn and needs previous training in Algol.

If we define the "intelligence" as the ability to make decisions due to a more sophisticated algorithms or equip the objects with some kind of artificial intelligence, only an advanced object-oriented algorithmic languages provide such features. Simula has this capacity. Perhaps this is the reason why Simula is still quite popular among the computer science researchers.

Anyway, if someone wants to create an object-oriented simulation package with intelligent objects, he/she finally must create a new high-level object-oriented algorithmic language. The question is: Isn't it better to take a known, complete, widely known, used, and advanced language and add to it the time and queuing management layer (TEM)?