Simulations I: Methodology

8

8.1 Introduction and Motivation

This chapter provides an introduction to social simulation as a major area of CSS research—independent, or almost independent, of the specific type of implementation. The core questions addressed in this chapter concern computer modeling and simulation in social science. Why use computer simulation as a methodology for scientific investigation of social complexity? The answer is—in brief—because formal theories of social complexity are sometimes more viable via computational modeling than through closed-form solutions. What unique insights on social complexity are gained through social simulation that are not available through other methodological approaches, such as statistical, mathematical, or historiographic? A major one is improved understanding of social complexity as an emergent phenomenon. What are the main limitations of social simulations? Full descriptions of social simulations are not as straightforward as thorough descriptions of other formal and statistical models, which sometimes can have significant consequences for replicating results. Another limitation is the relative shelf life of computer code as compared to mathematical models.

The main motivation for social simulation is based on the first two of these questions. Social simulations are capable of representing social systems and coupled socio-techno-natural systems in ways that other methodological approaches are not. Computer code in a well-chosen programming language or simulation system—such as those discussed in this and the next two chapters—provides a powerful formalism for theorizing, experimenting, and ultimately understanding social complexity.

8.2 History and First Pioneers

The following is a brief history of milestones and pioneers of social simulation research in CSS, with main emphasis on methodological concepts, principles, and practice—especially the founders' generation. Similar sections in the next two chap-

C. Cioffi-Revilla, *Introduction to Computational Social Science*,
Texts in Computer Science, DOI 10.1007/978-1-4471-5661-1_8,

ters focus more specifically on models. Some overlap between these summaries is unavoidable, since they are not completely disjunct.

1959 Oliver Benson at the University of Oklahoma pioneers the methodology of computer simulation in political science with his Simple Diplomatic Game Model.

1961–1971 Jay Forrester, founder of the System Dynamics Group at MIT, establishes the methodology of system dynamics theory and research through his classic monographs: *Industrial Dynamics*, *Principles of Systems*, *Urban Dynamics*, and *World Dynamics*.

1962 Psychologist and information science pioneer Harold Borko [1922–2012] publishes the edited volume *Computer Applications in the Behavioral Sciences*, possibly the first of its kind, including Julian Feldman's seminal chapter on "Computer Simulation of Cognitive Processes", Sydney and Beatrice Rome's computer simulation of large organizations, R. Clay Sprowls's "Business Simulation", and Benson's model.

1963 Political scientist Karl W. Deutsch [1912–1992] publishes *The Nerves of Government: Models of Political Communication and Control*, pioneering the information-processing paradigm of CSS, as a precursor to Simon's work. The same year Harold Guetzkow and collaborators publish the influential *Simulation in International Relations: Developments for Research and Teaching*, which soon becomes the new frontier.

1968 The Club of Rome, a major promotor of global carrying capacity modeling and simulation, is founded by Italian industrialist Aurelio Peccei and Scottish scientist Alexander King.

1969 Political scientists Hayward Alker and Ron Brunner publish the first comparative analysis of social simulation models in the journal *International Studies Quarterly*.

1970 Computer scientist James E. Doran publishes one of the earliest papers on the application of simulation methodology to archaeology, "Systems Theory, Computer Simulations and Archaeology", in the first volume of the journal *World Archaeology*.

1970s In Europe, social scientist Urs Luterbacher and collaborators at the Graduate Institute of International Studies in Geneva develop SIMPEST, the first numerical simulation model of political, economic, and strategic interactions based on a dynamical system of integral-differential equations, implemented in MINUIT. This model of the US-USSR-PRC triad correctly predicted the fall of the Soviet Union in late 1980s.

1970s In America, economist and strategist Thomas Schelling establishes foundations for a new methodological chapter in social simulations via cellular automata, and eventually agent-based modeling, through his study of racial segregation. John Casti, who later joined the Santa Fe Institute, coded the first implementation of Schelling's model while the two were at The Rand Corporation.

1972 Springer publishes the first edited volume on CSS in Europe, by Lucien Kern and collaborators, entitled *Simulation internationaler prozesse*, containing Jeffrey Krend's chapter on a replication of Oliver Benson's pioneering model.

1977 CSS pioneer Stuart Bremer [1943–2002] advances the methodology of social simulation with *Simulated Worlds: A Computer Model of National Decision Making*, published by Princeton University Press.

1980s Computer scientist Christopher Langton coins the term "artificial life".

1999 Computational social scientists Nigel Gilbert and Klaus Troiztch publish the first edition of the influential textbook, *Simulation for Social Scientists*.

2013 Computational social scientists Bruce Edmonds and Ruth Meyer edit the 754-page comprehensive handbook, *Simulating Social Complexity* by Springer. The same year both Springer and Wiley inaugurate specific series on Computational Social Science.

8.3 Purpose of Simulation: Investigating Social Complexity Via Virtual Worlds

The core scientific purpose of social simulation modeling and analysis is to investigate social complexity in ways that go beyond—often *way beyond!*—what is possible using other methodologies, such as historical, ethnographic, statistical, or mathematical approaches. This is accomplished by building a computer model of the social system or process under investigation—a virtual world representing relevant aspects of reality—and using that model to perform many kinds of analyses, as detailed in this and the next two chapters.

Reasons for using virtual worlds that simulate social complexity are numerous, including but not limited to the following:

Versatility: Many more complex social systems and processes can be investigated through simulation than through statistical or mathematical modeling. While every statistical or mathematical model can be simulated, the inverse is not true. Not every simulation model can be represented in mathematical form.[1]

High dimensionality: A common feature of social complexity, as we have seen in previous chapters, is having to analyze large numbers of variables, and interactions among them, a property called high-dimensionality. For example, emergence of collective action is a process involving numerous entities and variables, including situational parameters, goals, leadership characteristics, and resources, among numerous others. High-dimensional systems are common across domains of social complexity.

Non-linearities: Dynamic interactions among social entities are often nonlinear, independent of their dimensionality. Simple, low-dimensional systems are sometimes amenable to closed-form solutions, but that is generally not the case

[1] This is obviously not a blank criticism of statistical and mathematical models, which continue to play an essential role in CSS, as already shown in previous chapters.

for complex systems with high-dimensionality and nonlinear dynamics. Human perceptions, interaction as a function of physical distance, and patterns of cooperation and conflict are examples of nonlinear interactions. Social simulations can handle complex nonlinear dynamics, bound only by computational resources (which keep increasing).

Coupled systems: Another distinctive feature of social complexity is coupling among human, natural, and artificial systems, which virtually always implies high-dimensionality and nonlinear interactions. Computer simulation models provide an effective *and* efficient way of representing coupled socio-natural-artificial systems, as we will examine. For example, a computer model can be used to represent coupled dynamics among social institutions, the biophysical world of a society, and critical infrastructure.

Stochasticity: Randomness is ubiquitous and consequential in social systems and processes, as we have already examined. Stochasticity also comes in many forms, as defined by probability distributions. Examining the effects of diverse stochastic dynamics—how they generate patterns of social complexity—is another major reason for using simulations.

Incompleteness: Social science is incomplete, in the sense that not all parts of the social universe are known with the same degree of completeness. Social simulations are also used for testing alternative theories to advance our understanding of real-world social complexity.

Experimentation: The experimental method is a cornerstone of all science, but running experiments on complex social systems is not feasible for numerous reasons, including practical and ethical. Experimentation is rendered feasible through social simulations, including all classical features of this approach: treatments, control groups, and many different experimental designs. For example, computational experiments can be used to explore and test hypotheses concerning aspects of collective action, group dynamics, and governance under various assumptions of governance and public issues.

Policy analysis: Computer simulations of social complexity enable forms of policy analysis that are not available through other methodologies, including analysis of so-called "wicked problems"—the hallmark of hard challenges in policy analysis. For example, economic policies to mitigate inflation can be analyzed by modeling various actions such as wage subsidies or price controls.

These are powerful and compelling reasons! Interestingly, most of them are the same for scientists in other domains who use simulations—including astronomy, biology, and chemistry, among others—"Science in the 21st century is computational", as computer scientist Peter Denning once remarked.

8.4 Basic Simulation Terminology

Social simulation research employs a rich technical vocabulary that includes native CSS terms as well as terminology from computational science, such as object-oriented modeling and programming, UML, and related formal languages. For now

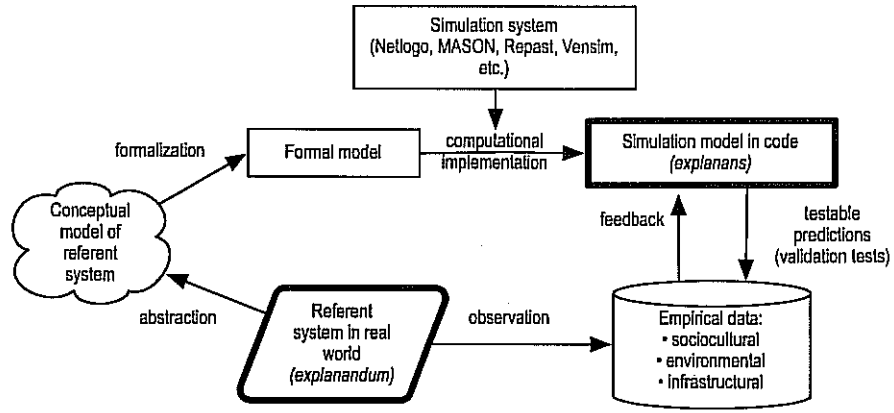

Fig. 8.1 *Basic terminology and general methodology of social simulation.* Social simulation methodology is an iterative process that begins with a referent system (*explanandum*) in the real world. Abstraction, formalization, programming, and appropriate data are used to develop a viable simulation model (*explanans*). This general process is independent of the specific kind of simulation model

we only need to clarify some initial terms; others will be presented as they are needed.

We shall use the following terms as *synonyms:*

- social simulation
- simulation model
- computer model
- machine simulation
- computational model
- simulated system

Hence, by "simulation", for short, we shall always refer to some kind of *computer model* of a social system or process, reserving the term "game" or "gaming" to human simulations solely based on role-playing.

The ontology of social simulation research includes the following basic terms, some of which are shared by other formal approaches, such as mathematical models. Consider Fig. 8.1, starting with the referent system (*explanadum*), in the bottom left and proceeding clockwise. Later we will use these initial building blocks to explain the methodology of modeling complex social systems as a systematic process.

Definition 8.1 (Referent System) A real-world system or process that is an object of investigation (*explanadum*) is called a referent system. Synonyms: target system, focal system, empirical or historical world.

Referent systems in CSS comprise the full *universe* of social entities, systems, and processes: the human mind, cognitive processes, decision-making, individual and group behavior, and societal, international, and global domains, including the World Wide Web. Some of the most complex referent systems in CSS are arguably

coupled socio-techno-natural systems, although a referent system of any degree of complexity may focus on a purely human/social system, or pairwise combinations of socio-technical and socio-natural subsystems.

A referent system is *defined* or *specified* by the specific *research questions* being investigated; it is not open-ended or all-inclusive, simply because it is located in the real-world. "Reality" is infinitely detailed and vast, objectively speaking. Scientific research always focuses attention on some selected subset of reality—i.e., a given referent system defined by research questions.

The following definition uses the term "abstract" as a verb to describe a key modeling activity.

Definition 8.2 (Abstraction) The process of selecting a given set of features from a referent system for modeling purposes is called abstraction.[2]

Thus, abstraction produces a simplified *conceptual representation* of the referent system, consisting of elements such as entities, variables/attributes, associations, and other patterns that provide specificity to the referent system being investigated. Sometimes the conceptual model is formalized into an intermediate mathematical model to better understand some properties of interest—as is typical in formal social theory.[3] The conceptual model is actually formalized into a simulation model when it is rendered in code. A simulation model may be written in native code, using one or more programming languages, or using some pre-existing simulation system.

Definition 8.3 (Simulation System) A computational toolkit or code library for building simulation models is called a simulation system.

A simulation system is a highly sophisticated computational artifact for building *other* advanced computational artifacts (specific models), which can be highly complex and inefficient/ineffective to build in native code. Netlogo, DYNAMO, Stella, Vensim, Swarm, MASON, Repast, and their predecessors, among many others, are examples of computational simulation systems. A social simulation *model* is to a simulation *system/toolkit* as a car is to a car factory; the former is made using the latter. You can also build a car on your own (good luck!), rather than buying one made in a factory—which would be the equivalent of writing a social simulation model in native code—but its performance and reliability will probably not come even close to a factory-made car. An important reason for using one of the latest existing simulation systems (Vensim, MASON, Repast, among others) is to reach levels of model performance and reliability that are unattainable by relying exclusively on purely native code. This is, emphatically, *not* an argument against building

[2]Note that the term "abstraction" has a different meaning in the context of computation, where it means hiding information, as discussed in Chap. 2.

[3]The full and powerful family of mathematical structures is available for this, including continuous, discrete, and hybrid formalisms.

simulation models; sometimes they are the best solution to a given set of research questions.

Multi-purpose computational mathematical systems, such as Mathematica and Matlab, are also used as simulation systems, to build and analyze models.

Some common (albeit not universal) facilities of simulation systems include the following:

Frequently used primitives: Code library of common primitives or basic building blocks for building a model. Examples: mathematical functions, distributions, simple agents, landscapes, schedulers, common data fields, constructor methods.

Random number generator: Simulation models require random number generators to represent processes, either substantive or procedural, with various forms of randomness (uniform, Poisson, power-law, among many others).

GUI: A graphic user interface is standard in most simulation systems, especially those intended for beginners and intermediate programmers, such as Netlogo, Repast, and Vensim.

Visualization tools: Used to draw histograms, time-series graphs, network diagrams, maps, and other visual aids for understanding simulation output.

More specialized facilities are usually added by model developers. These might include, for example, autocorrelograms and spectral diagrams, difference maps, heat maps, dynamic networks, Lorenz-curve graphs, and various non-Cartesian coordinate systems (e.g., spherical, cylindrical). All major simulation systems today have active user communities and some hold regular conferences or workshops.

Finally, a simulation model is implemented in code (*explanans*), as highlighted in Fig. 8.1, in the upper right, diagonally opposite the referent system (*explanandum*).

Definition 8.4 (Simulation Model) A model of a referent system that is formalized by code written in a given computer programming language (native or toolkit) is called a simulation model.

In the next chapters we will discuss different types of simulation models and examples of each. To do so in a systematic way, however, it is necessary to develop a viable classification of simulation models, given how many exist.

Figure 8.1 and the preceding definitions provide a first, high-level pass through the general methodology of simulation research in CSS. A more in-depth presentation is necessary, but several other distinctions are needed before delving into methodological details of actual simulation development or model construction.

8.5 Fidelity of Representation and Implications

Social simulations differ by the fidelity with which the computational model attempts to replicate or resemble a given referent system. The following ordinal scale distinguishes social simulations by increasing level of empirical specificity, which approximately follows a pure-applied science continuum:

1. At one end of the basic-applied continuum are highly abstract simulations that bear only sparse qualitative resemblance to a referent system, without attempt to replicate any quantitative features at all. Theoretical analysis as basic science is the main use of these models, not operational policy analysis.
2. At the next level toward "the plane of empirical observation" of the referent system—as philosopher of science Carl G. Hempel would have said—are simulation models that show convincing qualitative fit and some quantitative calibration. These models are still mostly theoretical, but they are capable of providing some applied insights. Since policies should not ignore basic science, findings from this class of social simulations may have valuable implications that policymakers ignore at their own peril. A good example of this is the classical Schelling segregation model (examined in Chap. 10), which is a rather abstract theoretical model that nonetheless sheds significant light on emergent patterns of social segregation and contributes key insights for policymakers.
3. Next are models with extensive qualitative and significant quantitative fit. This class of social simulations is of maximal interest for conducting empirically grounded CSS research. We shall examine several examples of this.
4. Finally, we come to social simulations that "look closest at the plane of observation" (in the sense of Hempel), such that quantitative and qualitative fit between simulation output and empirical data is the closest. High-fidelity simulations are calibrated to a referent system along multiple dimensions, which can be spatial (including numerous and detailed geographic features, down to a given scale of resolution, rendered through GIS and remote sensing data), temporal (defined to small time increments, such as decades, years, seasons, months, weeks, days, hours, minutes, and so on, down to the smallest scale of interest), or organizational (matching detailed network patterns at node, subgraph, and graph levels of analysis), among the most universal. Relatively fewer of these models are found in an academic context, but they are abundant in business and governmental organizations.[4]

This scale is totally unrelated to the merits or value of a simulation model, which is a different matter that has to do with scientific quality.[5] The fidelity scale is merely a heuristic way to locate a simulation model along a realistic-abstract continuum in order to understand its value and limitations.

There are numerous implications that follow from a model's representational fidelity. Perhaps the most obvious is that a simulation at one level cannot be expected to perform well at a different level. Thus, operational, high-fidelity models may have

[4]Part of the reason for this is that operational, high-fidelity models often require sensitive or proprietary information not normally used in academic CSS research.

[5]DARPA—the Defense Advanced Research Projects Agency of the US Department of Defense—uses a scale for classifying projects, ranging from "basic science" (called "6.1 projects", named so after the section in the relevant law) to more applied and operational research, labeled 6.2, 6.3, 6.4, etc., all the way up to fully operational systems deployed in the field for combat or humanitarian missions. The 6.X nomenclature is helpful and commonly used by other agencies.

significant policy value, but have little or no theoretical interest. Conversely, theoretical models can provide deep scientific insights and understanding, but offer little by way of actionable results as far as policy contributions are concerned.

A somewhat less obvious implication of the fidelity scale is that CSS researchers must make an effort to clarify as best as possible the desirable resolution of a model, *given the research questions*.

8.6 Types of Social Simulation: From System Dynamics to Agent-Based Models

Social simulation models constitute several major superclasses, the two largest being **variable-oriented models** and **object-oriented models**, with a third superclass of **hybrid social simulations** at their intersection. In turn, each superclass encompasses several significant classes, which can be characterized as follows. (Each class is examined in the next two chapters.)

Variable-based social simulations use systems of mathematical equations to implement the conceptual model abstracted from the referent system of interest. Historically, these were the earliest forms of simulations in CSS. **System dynamics simulations** and **queuing models** constitute major classes, both based on variables and deterministic or stochastic systems of equations for representing dynamic interactions.[6] The most distinctive feature of a system dynamics model (or SD, for short) is the representation of the state and dynamics of the referent system in terms of *levels* and *rates*, or "stocks and flows", respectively, in the form of a system of difference equations in discrete time. Hence, social systems that are abstracted as networks of states and rates of change are eminently suitable to this kind of simulation model. An SD system may be completely deterministic or partly stochastic.

A queuing model is more appropriate for rendering a referent system that receives some stream of inputs and releases the entities after some processing. The iconic example of this is a commercial bank, where customers arrive and wait in line while those ahead get served and depart the bank when they are finished. These models are stochastic, because waiting time and service time are generally stochastic, not deterministic. Accordingly, *probability distributions* play a major role in this class of social simulations.

These two classes of models are called variable-oriented because the modeling orientation upon which the abstraction is based looks first at the identification of key variables, such as levels of some stock and waiting time in a queue. Neither of these two classes of simulation models makes an effort to render the social entities (actors) explicitly; they are simply implied by state equations.

By contrast, object-oriented simulation models are based on an abstraction strategy that looks first of all at entities in the referent system. **Cellular automata** social simulations (or CA models, for short) consist of cells related to each other by neighboring relations on a landscape, such as in a city grid consisting of blocks, or a

[6]Note the exact terminology: "system dynamics", *not* systems dynamics (both plural) or dynamical systems (which refer to systems of differential equations).

patchwork of farms in the country. CA models look first at entities—the cells and their topology—and then at attributes/variables. Agent-based models are somewhat similar, as detailed in Chap. 10.

8.7 Development Methodology of Social Simulations

All social simulations, whether simple or complex, abstract or empirical, variable-oriented or object-oriented, are developed by systematic steps that begin with some core research motivation and end with a viable model. Although the specifics of each class sometimes matter, in general all social simulations follow a similar developmental methodology.[7] This section provides a second pass (spiral) through the cycle in Fig. 8.1.

8.7.1 Motivation: What Are the Research Questions Addressed by a Given Model?

The first step in social simulation modeling consists of careful formulation of viable research questions. Every social simulation is intended to address one or more research questions defined in terms of the referent system. In fact, a referent system is in large part defined by research questions; there is a synergistic relationship between the two. In an abstract SD model of inter-group rivalry the research questions may concern phase portraits and qualitative dynamical features. The same kind of model calibrated with historical data would be able to address research questions on the timing and magnitude of real-world conflicts. Similarly, research questions in an agent-based model will vary by level of fidelity, ranging from abstract, theoretical questions that may have to do with thresholds, elasticities, gradient fields, and similar theoretical concepts, to empirically referenced questions that might concern specific locations, actors, parameter values, or historical epochs.

Since research questions are a major engine for scientific inquiry, they largely define the level of fidelity and, therefore, also the scope of the referent system to be investigated. That being said, practical considerations may affect decisions on exactly how research questions are formulated.

- The relevant social science may be incomplete, so research questions may require adjustment in order to gain scientific coherence. The same is true for incompleteness in natural science or technology when modeling coupled referent systems.
- Empirical data necessary for initial research questions may be incomplete, poor, or downright nonexistent. This is a common situation in CSS research because researchers often pose questions that are tractable through computational tools, but no one has collected data necessary to verify or validate the models, thereby requiring adjustments to obtain viable research questions.

[7]The same is generally true of mathematical social science models, and also to some degree of econometric and other statistical models.

- Computational resources may be insufficient for an original set of research questions. This is another common occurrence, especially for overly ambitious projects that fail to estimate the correct amount or types of computational resources. This too usually requires limiting the scope of research questions asked.
- Other practical considerations, such as deadlines, and available personnel, may also condition the formulation of research questions.

The non-computational literature in social science may or may not provide adequate guidance in terms of research questions. This is because the computational approach in general, and the social complexity paradigm in particular, offer different human and social dynamics that are invisible from the perspective of non-computational literature. For example, vast areas of social science are practically defined in terms of a single methodology, such as statistical multivariate models, or game theory models, or general equilibrium models. By contrast, social simulation models address research questions that require any combination of formalisms. That being said, CSS researchers would do well in seeking to address research questions that are recognized as significant by non-computational scientists, as well as other CSS researchers.

Failure to begin with clear and viable research questions guarantees that subsequent complications will require backtracking until proper research questions are posed. This is sometimes inevitable, especially when new territory is being explored. However, such false starts should be avoided when possible, because they can be wasteful along multiple dimensions: time, costs, personnel, and missed opportunities. Scientific discipline and experience are valuable assets in the formulation of research questions in CSS, as in all domains.

A remark on interdisciplinary research in CSS: Research questions addressed through social simulations are frequently interdisciplinary because of multiple reasons. Social complexity respects no disciplinary boundaries! Coupled systems are multidisciplinary by definition. Complex social simulations, in particular, require interdisciplinary research.

8.7.2 Conceptual Design: What Does the Abstraction Look Like?

Given a set of viable research questions, the next step in developing a social simulation is to conduct a process of abstraction that will yield a conceptual model of the referent system. The abstraction itself should be informed and guided by the research questions.

Ideally, the abstraction for producing a conceptual model of a referent system should be guided exclusively by research questions and conducted without regard to consideration of subsequent implementation.

In practice, the abstraction and resulting conceptual model will be influenced by the known implementation resources. This is the tyranny of a hammer looking only for nails. If you know or use only method M, then both abstraction and resulting conceptual model will be shaped (and perhaps completely determined) by M, rather than by research questions, as it should be.

This methodological pathology in CSS research is similar to what happens in non-computational social science when researchers conduct abstractions and produce conceptual models guided primarily by those methods they know or prefer, rather than by what the research questions actually require. This methodological error should be avoided by gaining familiarity with different simulation approaches and a broad range of human and social phenomena—not easy, but well worth it. The abstraction and resulting conceptual model should contribute to answering the research questions, no matter what tools are required.

There is a history lesson to be learned here. A *major* source of methodological innovation comes from *not* having the proper computational tools to answer research questions. Isaac Newton was led to the invention of infinitesimal calculus because he wished to answer research questions for which there were no tools. He refused to adapt the research questions to existing tools or provide only tool-driven answers (like everyone else was trying to do). Likewise, John von Newman did the same by inventing game theory; he wanted to answer research questions having to do with interdependent choices (strategic entanglement), and the extant theory of decisions established by Bayes for answering questions of choice against nature was insufficient. Like Newton and others before him, he became a mathematician, invented game theory as a novel branch of mathematics, and then returned to the social science of interdependent decision-making and formalized it through game-theoretic models. He also invented cellular automata, examined in Chap. 10, which we now use for developing a broad class of social simulations. Simulation systems—from DYNAMO to MASON—were invented with the same science motivation: to enable us to expand scientific frontiers by answering an increasing number of challenging questions.

Different graphic systems have been invented to facilitate specification of a conceptual model. Flowcharts, Forrester diagrams, and UML diagrams are some examples. These are useful for refining ideas and they are indispensable in interdisciplinary projects when specialists from various domains need to develop consensus and common understanding. They will be examined in the context of each model class. No doubt, others will be invented as CSS research increases demand to create clearer conceptual models.

8.7.3 Implementation: How Is the Abstracted Model Written in Code?

The third step in developing a social simulation involves implementing the conceptual model into code. This is where a major decision is made in terms of implementing the conceptual model using native code or a simulation system such as one of those mentioned earlier. The choice is based on multiple considerations, which should include:

Research question Again, research questions should inform implementation, not just the conceptual model. The character of research questions and the resulting conceptual model should first determine whether the simulation model should be variable-oriented (attributes are most prominent) or object-oriented (entities are most prominent) and, second, whether native code or a toolkit should be used.

Expertise Excellence in some implementation solutions may also bring novel answers to research questions. For example, a CSS team highly skilled in building SD models can make significant contributions to a given domain, even if alternative OO (object-oriented) models are possible. Different formalisms of the same referent system almost always bring to light different aspects that advance understanding.

Future use Consideration should be given to future uses that may be envisioned. Such uses include further research, use in teaching, or policy analysis or problem-solving.

The main result of this third step is an *initial version* of a simulation model, which will likely evolve through subsequent *versions*. By convention, the *initial version* of a simulation model is labeled 0.1 or lower. Relatively small, incremental changes prompt *decimal* increases in version numbers, whereas relatively large or major changes prompt *integer* increases—a protocol similar to numbering versions of "the same" software. In general, there are more decimal increases than integer increases.

A social simulation implemented in code should abide by all the principles discussed in Chap. 2 concerning best practices, such as commenting, modularity, defensive programing, multiple backups, and similar guidelines. Code that can no longer be understood even a year after it was written is useless.

In all cases, model code must be committed to some depository. Sourceforce, Googlecode, the Harvard-MIT Data Center (Dataverse), and OpenABM provide examples of online, open-source, code depositories. Besides code files, documentation must also be provided, including all supplementary supporting files. A great deal of effort goes into producing a high-quality model, as we will discuss later in this chapter. However, *simulation code is highly perishable, far more so than mathematical or statistical models*. Unfortunately, it is not uncommon for social simulations—even famous ones—to be lost within a relatively short span of time following their creation. Often all that remains is the conceptual model and some mathematical features.

8.7.4 Verification: Does the Simulation Perform as Intended?

The process of finding out whether a simulation model is working as intended by the conceptual model is known as **verification**, a procedure that also involves **debugging**. This is equivalent to what is traditionally called *internal validity* in non-computational social science formal methodology. An unverified model cannot be used for analysis. Verification is accomplished through multiple procedures, as detailed below. All of them typically unveil bugs hidden in the initial simulation code.

8.7.4.1 Code Walk-Through

Reading code line by line, commenting and refactoring it as necessary, is an indispensable procedure to ensure a simulation is working as intended by both model designers and programmers. Modularization facilitates this procedure, as well as providing other benefits. Code walk-through (also written as walkthrough) should be done while also consulting all relevant prior documentation, including conceptual narratives and diagrams. Again, good programming style resulting from best practices facilitates the code walk-through procedure.

8.7.4.2 Profiling

Another procedure for verifying code is to "profile" it. Profiling means to count the frequency with which key code elements are used, such as various methods or operations in OOP (object-oriented programming) code, or functions in other programming languages. In a sense, profiling is a form of quantitative, automated content analysis or information extraction procedure conducted on code—a means of *mining code* to detect possible errors. The result of profiling is a quantitative summary of findings, such as a frequency histogram of methods or functions called. Formally, the result of profiling code is a rank-size distribution, which resembles the idea behind a Type I Zipfian power-law model. Often it is impossible to draw inferences on the sole basis of profiling results; however, when added to other information from code walk-through, profiling can be a valuable procedure.

8.7.4.3 Parameter Sweeps

Social simulation models typically include large numbers of parameters. Such a large set of space parameters can be used for verification purposes by evaluating the model as a single parameter changes in values while others are held constant. Thus, results from a parameter sweep will provide a response surface which can be plotted and examined for possible anomalies indicative of bugs or other patterns that should not appear. Parameter sweeps can reveal special properties within a range, such as singularities, asymptotic behaviors, oscillations, or other quantitative and qualitative patterns.

8.7.5 Validation: Can We Trust the Results?

The process of finding out whether results from simulation model runs match what is known from empirical data is known as **validation**. Essentially, validation involves pattern matching between simulation output and observed patterns in the referent system.

There are a variety of ways in which simulation validation is conducted. Among the most important and common ones are:

Histograms: Frequency distributions obtained from simulation runs can be matched with empirical histograms—for example, income distributions, the size of spatial distributions, and similar.

Distribution moments: All distributions are characterized by moments, so matching moments generated by simulation runs with real data is another strategy.

Time series: Dynamic social simulations typically produce time-series data from simulation runs, which can be compared with empirical time series.

Special indices: Specific measures, such as the Gini coefficient, entropy, the Hurst coefficient, and similar indices can also be used.

Other: Results from simulation runs produce numerous statistics and patterns that are often characterized by the specific subject matter and can be used to compare with real-world data.

Sometimes an existing simulation system, such as Netlogo, MASON, or Repast, will already have some of these facilities for conducting model validation tests. However, it may be necessary to develop such facilities in the case of frequently used validation tests that are not provided by the simulation system being used.

Ideally, validating a social simulation model is facilitated by pre-existing empirical data that can be used to match results from simulation runs. This is often the case when data from simulation runs also exists in reference to actual empirical data. However, it is not uncommon to discover that simulation results produce data that has never been measured in the real world. In this case, there is no choice but to attempt to collect additional data as necessary. An interesting scientific situation arises when a social simulation produces results that no one has looked for before!

Validating a social simulation model also involves estimating and calibrating parameter values to their appropriate ranges. This is often done by beginning with existing empirical parameter values or informed guesses within a justifiable domain. In the end, validation always involves matching simulated, virtual data, with real, empirical data.

8.7.6 Virtual Experiments and Scenario Analyses: What New Information Does the Simulation Generate?

Earlier we discussed how virtual experiments are a major scientific contribution of social simulation models. Conducting virtual experiments, such as by analyzing alternative scenarios, is an intriguing and exciting use of computational modeling.

Computational experiments using social simulation models can be based on basic scientific research, as well as on applied policy analysis. Analyzing virtual experiments and alternative scenarios is a social simulation tradition that goes back to the earliest days of computer simulation modeling in the social and behavioral sciences. For example, the earliest system dynamics global models were used to analyze industrial development policies and global environmental trends under a variety of future scenarios. While many of the assumptions used in these initial models during the 1970s proved to be incorrect, the methodology itself was powerful and continues to develop to this day.

Conducting virtual experiments through simulation models is also common in other computational disciplines ranging from biology to astronomy. The reason for this affinity between CSS and computational biophysics and the earth and space

sciences is the common problem of being unable to conduct real experiments on the referent systems of interest. The only way to understand what happens when two galaxies collide is to conduct computational experiments, much the same as is the case for conducting virtual experiments in computational biology.

8.8 Assessing the Quality of a Social Simulation

Social simulation methodology has begun to generate proposals for assessing and promoting quality across diverse and related areas.[8] For instance, proposals exist in the area of communicating social simulation models, assessing complex projects that involve large interdisciplinary teams (Sect. 8.9), and comparing models (see Sect. 8.10). A strong consensus on a universal set of quality standards in social simulation research has not yet emerged, but such a debate has already begun in the global CSS community.

8.8.1 General Principles for Social Modeling Assessment

The criteria of "Truth", "Beauty", and "Justice" have been proposed by Charles A. Lave and James G. March in the classic *Introduction to Models in the Social Sciences* (1993). These criteria are widely used for discerning quality in *social science formal models*, mainly mathematical in kind. The three terms "Truth", "Beauty", and "Justice" (or TBJ, for short) are labels for quality dimensions referring to fundamentally good—i.e., normatively desirable—features of social science modeling. Accordingly, the TBJ terms must be interpreted not literally but as labels.

Truth refers to the empirical explanatory content of a model—i.e., its contribution to improving causal understanding of social phenomena—in the sense of developing positive theory. For example, truth is normally judged by internal and external validation procedures, corresponding to axiomatic coherence and empirical veracity, respectively. Truthfulness is the main, classical criterion for evaluating empirical science, whether a model is statistical, mathematical, or computational. Truth must be a constituent feature in a social science model; without it, a model has no overall quality contribution.

Beauty refers to the esthetic quality of a model, to its elegance in terms of properties such as parsimony, formal style, syntactical structure, and similar features. Beauty is about art and form. For example, the mathematical beauty of some equations falls within this criterion, including features such as the style of a well-annotated system of equations where notation is clear, well-defined, and elegant.

[8]This section focuses on social simulations, so the broader field of CSS (e.g., social data algorithms or socioinformatics, complexity models, social networks, social GIS, and related areas of social computing) lies beyond the scope of this section. Quality research in those other areas is subject to its own standards, as discussed in previous chapter.

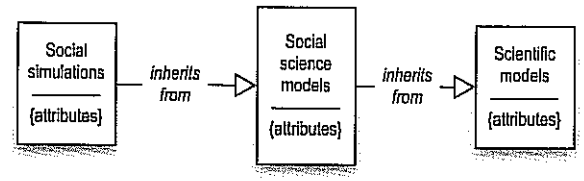

Fig. 8.2 UML class diagram illustrating the hierarchy of scientific models (*left*), social science models (*center*), and social simulations (*right*), each having increasingly specific standards for judging quality (moving from *left* to *right*). *Source*: Cioffi-Revilla (2013)

Unlike truth, beauty is not necessarily a constituent attribute, but is certainly a desirable scientific quality.

Justice refers to the extent to which a model contributes to a better world—to improvement in the quality of life, the betterment of the human condition, or the mitigation of unfairness. Justice is a normative criterion, unlike the other two that are positive and esthetic. For example, a model may improve our understanding of human conflict, inequality, refugee flows, or miscommunication, thereby helping to mitigate or improve social relations and well-being through conflict resolution, poverty reduction, humanitarian assistance, or improved cross-cultural communication, respectively. Policy analysis can be improved by social simulation models that are properly validated.

These Lave-March criteria of truth, beauty, and justice are useful for evaluating the quality of social simulation models. For example, in the classic Schelling model of segregation all three criteria are well-recognized. This is a fundamental reason why Schelling's model is so highly appreciated.

However, a further challenge exists because social simulations have features that render truth, beauty, and justice insufficient as criteria for assessing quality. This is because social simulation models are instantiated or rendered in code (a computer program in some language), so one can easily imagine a social simulation that would be of high quality in terms of truth, beauty, and justice, but fail in overall quality because simulation models pose additional challenges beyond other social science models (i.e., beyond the features of statistical or mathematical models).

As illustrated in Fig. 8.2, social simulations have properties that are shared with all models in science generally and social science in particular, based on inheritance as a specialized class, in addition to having other features of their own. For example, the specific programming language of an agent-based model (Java, C++, or other), or that of a system dynamics model, would be a defining feature.

The inheritance relation between social science models and social simulations readily suggests several key features that distinguish the latter from the former, as illustrated in Table 8.1.

Additional criteria for social simulations—i.e., criteria beyond classical standards for social science models—should allow us to judge quality in terms of "The Good, The Bad, and The Ugly".

Table 8.1 Quality criteria for evaluating models in domains of science

Models in ...	Truth	Beauty	Justice	Additional criteria
Science	Yes	Yes	No	No
Social science	Yes	Yes	Yes	No
Social simulation	Yes	Yes	Yes	Yes

Source: Cioffi-Revilla (2013)

Common required practices, such as verification and validation, are well-known quality control procedures for assessing scientific models in general. However, verification and validation are insufficient criteria for assessing the quality of social science models, specifically for social simulations. An important implication is that current emphasis on model verification and validation is warranted, but *verification and validation are insufficient by themselves for judging the quality of a social simulation model* (agent-based or other).

Therefore, a key methodological question concerning quality is: which additional criteria—i.e., beyond truth, beauty, and justice—could or should be used to assess the quality of a social simulation model? We shall now address this question based on a set of dimensions for evaluating the quality of a given social simulation model.

8.8.2 Dimensions of Quality in Social Simulation Models

The quality of any complex artifact—whether a social simulation model or the International Space Station—is a multifaceted property, not a single dimension. Dimensions of quality can be used for evaluation and can also provide a master checklist of desirable attributes for building and developing a social simulation model. Arguably, there are two levels of quality assessment for computational social simulations corresponding to the concepts of *a model* and *modeling*, respectively.

First, from a model's perspective, any set of quality dimensions for evaluating a social simulation must be based on its specific attributes or uniquely constituent features as a computational artifact in the sense of Simon. Moreover, whether the overall quality of a given model should be an additive or a multiplicative function of individual qualitative features is less important than the idea that overall quality depends on a set of dimensions or desirable features beyond the Lave-March criteria, not on some single preeminent feature (e.g., simulation environment or programming language).

Second, from a modeling perspective, quality assessment should cover the broader modeling or model-building process as such, beyond the social simulation model that is produced in a narrow sense. This is because a computational model in final (i.e., committed) instantiated code is the result of a sequence of earlier modeling stages that precede the model itself, such as the critical stage of model design prior to implementation. *Quality in design affects quality in the product of implementation*, even when implementation *per se* is carried out in a proper manner (i.e., competently, with effectiveness and efficiency).

The following Lifecycle Framework for quality assessment combines both perspectives—the model and its developmental process—by focusing on the classical methodological stages of social simulation modeling, as we discussed earlier in this chapter, with only minor modifications:

1. Formulation
2. Implementation
3. Verification
4. Validation
5. Analysis
6. Dissemination

Such a framework provides a viable checklist of quality dimensions to consider, based on the preceding methodological principles for social simulation research. Note that verification and validation constitute only two contexts for assessing quality and, as shown below, some of the others involve quite a number of *additional aspects* regarding quality evaluation.

1. **Formulation**. Quality can be assessed starting from the formulation of a research problem that a given social simulation is supposed to solve. A first set of quality assessments regards research questions. Is the research question or class of research questions clearly formulated? Is the focal or referent empirical system well-defined? Beyond clarity, is the research question original and significant? Originality should be supported by complete and reasoned surveys of prior, extant literature to assess scientific progress. Every computational simulation model is designed to address a research question, so clarity, originality, and significance are critical. Motivation is a related aspect of problem formulation. Is the model properly motivated in terms of relevant extant literature? Or, is the simulation model the very first of its kind? If so, are there prior statistical or mathematical models in the same domain? Literature reviews in published social simulation research should not be incomplete, poorly argued, or totally missing.
2. **Implementation**. Rendering an abstracted model in code involves numerous aspects with quality-related implications, starting with aspects of instantiation selection. Does the code instantiate relevant social theory? Is the underlying social theory instantiated using a proper program or programming language? Code quality brings up other aspects that may be collectively referred to as the Grimson-Guttag standards: Is the code well-written? Is the style safe/defensive? Is it properly commented? Can it be understood with clarity one year after it was written? In addition, what type of implementation strategy is used? I.e., is the model written in native code or using a toolkit? If a toolkit is used, which one, why, and how good is the application? Is the choice of code (native or toolkit) well-justified, given the research questions? In terms of "nuts and bolts", quality questions include such things as: What is the quality of the random number generator (RNG)? Is it Mersenne Twister, MT19937, or other PRNG? Which types of data structures are used, given the semantics? Are driven-threshold dynamics used? If so, how are the firing functions specified? In terms of algorithmic efficiency, what is the implementation difficulty of the problem(s) being addressed by the model? How efficient is the code in terms of implementing the main design ideas? In terms of computational efficiency, how efficient is the code in

terms of using computational resources? This aspect differs from algorithm efficiency. From the perspective of architectural design, is the code structured in a proper and elegant manner commensurate with the research question? In terms of object ontology, does the model instantiate the object-based ontology of the focal system for the chosen level of abstraction? Note that all these quality-related questions precede verification and validation.

3. **Verification**. Which passive and active tests were conducted to verify that the model is behaving in the way it is intended to behave? Social scientists also call this internal validity. Verification tests include but are not limited to the following: code walk-through, debugging, unit testing, profiling, and other common procedures used in software development, as we have already seen, and will examine more closely in the next chapters. What were the results of such verification tests? Quality assessment should cover investigation of which verification procedures were used, since results can range widely depending on the extent of verification methods employed. Unfortunately, most social simulations are reported without much (or any) information regarding verification procedures, as if it were true that "results speak for themselves"—quite often they do not.
4. **Validation**. Similarly, validation of a social simulation, what social scientists call external validation (or establishing a model's external validity), consists of a suite of tests, not a single procedure. Such tests are important for assessing quality in a social simulation. Which tests (histograms, RMSE for assessing goodness of fit, time series, spatial analysis, network structures, and other forms of real vs. artificial pattern matching tests) were conducted to validate the model? What were the results? Validation tests are often the focus of reporting results at the expense of all other phases in the life cycle of a social simulation model.
5. **Analysis**. The preceding aspects provide a basis for establishing overall confidence in a given model. What is the level of confidence in the model's results, given the combined set of verification and validation tests? If networks are present and significant in the focal system, does the model exploit theory and research in social network analysis (Chap. 4)? Does the model facilitate analysis of complexity as a system of non-linear interactions and emergent properties (Chap. 6)? Which features of complexity (emergence, phase transitions, power-laws or other heavy-tailed distributions, criticality, long-range dynamics, near-decomposability, serial-parallel systems, or other structural features) are relevant to the particular model? If spatial features are significant, does the simulation employ appropriate spatial metrics and statistical tools for spatial data? What is the overall analytical plan in terms of simulation runs and how is it justified? How does computational analysis advance fundamental or applied understanding of social systems? In terms of overall effectiveness, does the model render what is necessary for answering the initial research question(s) or class of research questions? This differs from efficiency. In terms of the simulation's computational facilities, does the model possess the necessary functionality for conducting extensive computational analysis to answer the research questions or even go beyond? How powerful is the model in terms of enabling critical or insightful

experiments, for example in terms of parameter exploration (evolutionary computation) and record-keeping? What is the quality of the physical infrastructure that renders the most effective simulation experience?

6. **Dissemination**. Finally, the quality of a social simulation should be assessed in terms of its "life-beyond-the-lab". For instance, in terms of pedagogical value: Does the model teach well; i.e., does it teach efficiently and effectively? In terms of communicative clarity and transparency, are useful flowcharts and diagrams of various kinds (e.g., UML class, sequence, state, and use case diagrams) provided for understanding the model? Are they drawn with graphic precision and proper style? In terms of replicability, what is the model's replication potential or feasibility? How is reproducibility facilitated? Aspects related to a model's graphics are also significant for assessing quality, not just "eye candy". In terms of GUI functionality, is the user interface of high quality according to its main users? Is the GUI foundational for answering the research questions? More specifically, in terms of visualization analytics, is visualization implemented according to high standards? This does not concern only visual quality, but analytics for drawing valid inferences as well. From a perspective of "long-term care", what is the quality of the model in terms of curatorial sustainability? How well is the model supported in terms of being easily available or accessible from a long-term perspective? In which venue (Google Code, Sourceforge, OpenABM, Harvard-MIT Data Center/Dataverse, or documentation archives such as the Social Science Research Network SSRN) is the model code and supplementary documentation made available? Finally, some social simulations are intended as policy analysis tools. Is the model properly accredited for use as a policy analysis tool, given the organizational mission and operational needs of the policy unit? Does the model add value to the overall quality of policy analysis? Does it provide new actionable information (new insights, plausible explanations, projections, margins of error, estimates, Bayesian updates) that may be useful to decision-makers?

The quality of a social simulation is proportional to the number of dimensions on which it is highly rated. Although these basic dimensions are not independent among themselves, their total contribution is what matters in terms of a comprehensive quality assessment.

8.9 Methodology of Complex Social Simulations

Some social simulations are called *toy models* because they represent a very simple referent system based on research questions that investigate a relatively narrow range of entities and dynamics. Some of the earliest social simulation models belong to this class, and they are still important today because they provide a unique way of understanding fundamental human and social dynamics. For example, toy models such as Heatbugs, Segregation, Hawks and Doves, or Boids—as well as many others provided by Netlogo—have significant pedagogical value for teaching the fundamentals of social simulation science.

Other models consist of **complex social simulations** and are characterized by numerous interacting entities, typically heterogenous in several respects, governed by multiple and typically nonlinear dynamics. Complex social simulations are normally built by interdisciplinary teams with distributed expertise among members. Typical cases in this group include coupled socio-techno-natural systems that require integrated application of knowledge across multiple domains. Such models also typically require years of development work, most often involving multiple research institutions.

The methodology of complex social simulation models requires special consideration in order to exploit the richness of such models while at the same time managing multiple challenges. A viable approach to complex social simulation modeling is to view model development as a spiraling, multi-stage process that proceeds from an initial, simple model and moves toward the much more complex final model. A famous example of this in the history of physical science was none other than Isaac Newton's research program on planetary dynamics (what prompted him to invent infinitesimal calculus), which has been studied in detail by the late Hungarian philosopher of science and mathematics, Imre Lakatos [1922–1974]. As described by Lakatos, Newton worked through a progressive sequence of models—not a single large model—before he arrived at his final, full model of the whole planetary system, complete with planets, moons, and the sun at its center. The initial simple model investigated by Newton bore no resemblance to the final model, except as a minuscule component. His first model consisted of a single perfect sphere rotating around its axis. Subsequent models in a cleverly chosen sequence of "progressive problemshifts" added moons, tilting axes of rotation, elliptical orbits, and numerous other carefully chosen empirical features as Newton approximated his final model of the planetary system. The entire movement from the initial, simple model to the final, complex model resembled the masterfully orchestrated music of Maurice Ravel's *Boléro*, which starts with a single, lonely drum and ends with a huge, full orchestra.

An example of a complex social stimulation, in many ways similar to Newton's final model of the planetary system, would be a coupled socio-techno-natural system. In order to develop such a simulation as a final model of a referent system representing some geographic region, the first initial model would represent a single territorial entity with minimal dynamics included in the simulation. Once such an initial model is well understood, additional features would be added. For example, the second model in the sequence would have heterogeneous agents, in order to understand more realistic cultural dynamics. A third model would add some simple weather dynamics, to further understand biophysical interactions between, say, precipitation and land cover used by agents. The fourth model could include multiple societies over a broader region. Subsequent models would add infrastructure systems and other technological artifacts.

The idea of a sequence of models for developing a complex simulation research program should not be misinterpreted as being a strictly linear process. Occasionally, it is necessary to make corrections and return to an earlier model that overlooked something important, or it may be necessary to develop deeper understanding

of simpler dynamics. That being said, the methodology of complex social simulations should have a definite forward thrust, moving from simple (initial model) to complex (final model).

There are several distinctive features of the methodology of complex simulations.

1. It is necessary to identify an initial model that is simple enough to understand in full detail, while at the same time representing a core element of the envisioned final model of the referent system. Note that the very first model may not bear much resemblance to empirical entities, just as in Newton's case a perfect sphere did not represent any real planet.
2. The sequence of models leading up to the final simulation is not arbitrary; it must be carefully designed in order to provide cumulative insights as work proceeds toward the final model. The sequence of simulation models should follow a theoretically meaningful plan, not simply proceed by random accretion and incremental changes without theoretical justification.
3. Verification is an essential activity throughout the whole development process from one model to the next. However, validation should proceed in a very judicious way, lagging behind verification, because if the model is tested through validation procedures that are premature with respect to the final model, what happens is that theoretically significant models might be rejected because they lack sufficient empirical support. This was the case with Newton's initial models in the sequence, which is why he was not as concerned with empirical tests early on in the research program.
4. Defining a final simulation model for the referent system is essential, because a progressive sequence of models can go on indefinitely.

Again, a clear focus on core research questions is essential for governing the development of complex simulations, just as it is for simpler models.

8.10 Comparing Simulations: How Are Computational Models Compared?

Comparative research is a well-developed and fruitful endeavor with a rich history across the social sciences. In fact, the theory and practice of comparative methodology is viewed by many as a defining feature of social science. Systematic comparison of social simulations is insightful and instructive for multiple reasons:

1. *Research questions* investigated through social simulation are clearly highlighted when comparing simulation models because research questions define the simulations themselves.
2. Analyzing *similarities and differences* among social simulation models provides a deeper, more comprehensive way of understanding them.
3. Comparative analysis of two or more social simulations can help identify features such as overlaps, gaps, or questions in need of *further research*.
4. Insights from comparative analysis of social simulations can also be used to clarify and refine *fundamental dynamics*, such as key properties of emergent phenomena in social complexity.

Keeping in mind the three main types of models used across the social sciences—i.e., statistical, mathematical, and computational varieties—it is safe to say that social scientists have learned a great deal from comparing statistical and mathematical models. For example, social scientists often compare various types of statistical regression models, such as when deciding which type to use given a set of hypotheses being tested, or when analyzing results from alternative functional specifications. Another example is provided by comparing game theoretic models, such as the classic taxonomy of 2 × 2 games pioneered by the late Russian-American mathematical social scientist Anatol Rapoport. Comparing social simulation models is a newer endeavor when compared to statistical and mathematical models.

A first approach to comparing social simulations is based on generic characteristics such as their referent system, type of implementation, level abstraction, and basic science versus applied uses. Each of these features provides ample room for examining similarities and differences among models being compared. Moreover, depending on the purpose of comparison, these features can be investigated in various degrees of detail. For instance, comparing social simulations by type of implementation is something that can be done in coarse terms by simply identifying the programming languages or simulation systems, or it can be much more detailed, comparing architectural features and interaction networks captured by each implementation. Comparison by generic characteristics can also focus on behavioral dynamics, distributions and stochastic processes, forms of emergent complexity, and long-term asymptotic equilibria.

The more advanced comparison of social simulations should focus on detailed examination of ontologies (including details provided in technical diagrams), dynamic processes (for example, by comparing UML sequence diagrams and state diagrams, in the case of agent-based models), as well as numerous other software features.

Comparing social simulation models is also sometimes referred to as **model-to-model** comparison, or **M2M** for short. In the next two chapters we shall examine several examples.

Recommended Readings

H.R. Alker Jr., R.D. Brunner, Simulating international conflict: a comparison of three approaches. Int. Stud. Q. **13**(1), 70–110 (1969)

O. Balci, Verification, validation, and testing, in *Handbook of Simulation*, vol. 10, (1998), pp. 335–393

J.L. Casti, *Would-Be Worlds: How Simulation is Changing the Frontiers of Science* (Wiley, New York, 1997)

C. Cioffi-Revilla, On the methodology of complex social simulations. J. Artif. Soc. Soc. Simul. 13(1), 7 (2010). Available online

C. Cioffi-Revilla, Comparing agent-based computational simulation models in cross-cultural research. Cross-Cult. Res. **45**(2), 1–23 (2011)

C. Cioffi-Revilla, Computer simulation, in *Leadership in Science and Technology: A Reference Handbook. Volume 1: General Principles*, ed. by W.S. Bainbridge (Sage, Thousand Oaks, 2012), pp. 345–354

B. Edmonds, R. Meyer (eds.), *Simulating Social Complexity* (Springer, Berlin, 2013)

W.G. Kennedy, C.R. Cotla, T. Gulden, M. Coletti, C. Cioffi-Revilla, Towards validating a model of households and societies of East Africa, in *Proceedings of the Fourth World Congress in Social Simulation (WCSS 2012)*, Taipei, Taiwan, Republic of China (2012)

C.A. Lave, J.G. March, *An Introduction to Models in the Social Sciences* (University Press of America, Lanham, 1993)

J. Rouchier, C. Cioffi-Revilla, J.G. Polhill, K. Takadama, Progress in model-to-model analysis. J. Artif. Soc. Soc. Simul. **11**(2–8) (2008)

R.G. Sargent, Verification and validation of simulation models, in *Proceedings of the 40th Winter Simulation Conference, WSC'09*, (2008), pp. 157–169. Available online

R.K. Sawyer, Social explanation and computational simulation. Philos. Explor. **7**(3), 219–231 (2004)

F. Squazzoni (ed.), *Epistemological Aspects of Computer Simulation in the Social Sciences* (Springer, Heidelberg, 2009)

Simulations II: Variable-Oriented Models

9

9.1 Introduction and Motivation

This chapter examines the superclass of *variable-oriented* social simulation models, also called *equation-based* social simulations. Historically, these were the first types of social simulations and they have formal roots in differential equation models of social dynamics. Today, these social simulation models consist primarily of **system dynamic (SD) models** and **queueing models**. Each class is examined using the MDIVVA social simulation methodology (Motivate-Design-Implement-Verify-Validate-Analyze) developed in Chap. 8.

Both of these social simulation models focus on complex social systems *over time*, which makes them applicable to theoretical application for basic science as well as policy analysis. Historically, however, applications to applied operational and management issues have prevailed. Hence, their use for advanced theoretical analysis awaits many fruitful applications, especially in light of experience acquired through practical uses in management, industrial, and operational settings.

9.2 History and First Pioneers

Social simulation models examined in this chapter have scientific roots in Isaac Newton's theory of dynamical systems and Girolamo Cardano's theory of events in probability—a prestigious pedigree. The following summary of major milestones includes developments in SD and queueing models as well as closely related advances in dynamic simulation models more broadly.

1909 Mathematician and engineer Agner Krarup Erlang pioneers scientific research on queuing systems by modeling the Copenhagen telephone exchange.

1953 Statistician and mathematician David G. Kendall proposes the standard formal notation still in use for queueing systems, published in *The Annals of Mathematical Statistics*.

C. Cioffi-Revilla, *Introduction to Computational Social Science*,
Texts in Computer Science, DOI 10.1007/978-1-4471-5661-1_9,

1958 Richard Bennett at MIT creates SIMPLE (Simulation of Industrial Management Problems with Lots of Equations), the first system dynamics computer modeling language.

1959 DYNAMO (DYNAmic MOdels) v. 1.0, an improved version of SIMPLE, is invented by Phyllis Fox and Alexander Pugh. DYNAMO quickly becomes the formal *lingua franca* of management science and operations simulation models.

1960s SD models become widely adopted in operations research of complex social systems and management science, remaining prominent today.

1961 Engineering scientist Jay Forrester from MIT's Sloan School of Management publishes his pioneering book, *Industrial Dynamics*, the first in a series of SD classics.

1961 Applied mathematician Thomas L. Saaty publishes the queueing theory classic, *Elements of Queueing Theory with Applications*. In the same year J.D.C. Little publishes his famous law of queueing systems in the journal *Operations Research*, and J.F.C. Kingman publishes his equally famous law in *Mathematical Proceedings of the Cambridge Philosophical Society*.

1969 *Urban Dynamics* is published by Jay Forrester and John Collins (former mayor of Boston), expanding system dynamics simulation to social complexity and CSS in a proper sense.

1970 Forrester and his group at MIT create the first socio-environmental global models, WORLD1 and WORLD2, published as *World Dynamics*, of what eventually became the famous Club of Rome model.

1972 *The Limits to Growth*, the classic book that will make SD famous worldwide, is published by Donella Meadows under the sponsorship of engineer Aurelio Peccei's Club of Rome. It is immediately translated into many languages.

1972 Cultural anthropologist Linda S. Cordell pioneers the first social simulation of Puebloan (Anasazi) polities in the American Southwest with her Ph.D. dissertation on "The Whetherill Mesa Simulation" at the University of California at Santa Barbara. Cordell received the Lifetime Achievement Award from the Society for American Archaeology and the A.V. Kidder Medal from the American Anthropological Association, becoming a member of the US National Academy of Sciences in 2005.

1975 Political scientist Dieter Ruloff, disciple of CSS pioneer Daniel Frei from the University of Zürich, Switzerland, demonstrates the first application of SD to simulating insurgency and political stability. In the following years he publishes the first SD models of the collapse of Classic Maya polities and Soviet–Taliban insurgency dynamics in Afghanistan.

1975 Political scientists Nazli Choucri and Robert North publish *Nations in Conflict*, the first discrete-time simulation in international relations, modeling the onset of World War I .

1979 Political scientists Urs Luterbacher and Pierre Allan from the Graduate Institute of International Studies in Geneva, Switzerland, create SIMPEST, the first dynamic simulation model of USA-USSR-PRC strategic triad dynamics during the Cold War, correctly predicting the disintegration of the Soviet Union. Their paper was presented at the World Congress of the International Political Science Association, Moscow, USSR.

1979 Archaeologists Colin Renfrew and K.L. Cooke co-edit the volume *Transformations: Mathematical Approaches to Culture Change*, another early pioneering collection.

1981 Archaeologist Jeremy Sabloff publishes *Simulation in Archaeology*, one of the first edited volumes of its kind. The same year Nazli Choucri publishes *International Energy Futures*, the first SD modeling book on the world energy market from an economic and politics perspective.

1984 The SD scientific journal, *System Dynamics Review*, is founded.

Mid-1980s Political scientist Michael Wallace publishes a paper demonstrating the implementation of Lewis F. Richardson's theory of arms races in SD models using DYNAMO.

1985 The Stella version 1.0 software for system dynamics modeling is released by the isee systems company.

1998 Nazli Choucri and her MIT students publish the first SD model of state stability in the *System Dynamics Review*.

2000 American management scientist John D. Sterman publishes *Business Dynamics: Systems Thinking and Modeling for a Complex World*, the first major, comprehensive textbook in SD.

9.3 System Dynamics Models

This section introduces the superclass of social simulations based on system dynamic (SD) models, used in significant social science applications, and examines their main features for understanding social complexity. SD models are introduced within the broader context of dynamical systems, which span an even larger class of formal models. The emphasis of SD is on discrete-time systems as the main formalism for characterizing social dynamics of various types observed in referent social systems. Mathematical aspects are important, especially for learning how qualitatively different dynamical processes—i.e., different forms of dynamic behavior—are modeled through different model specifications.

The following terms must be distinguished in the interest of clarity, since they are easily confused when not used with precision:

Definition 9.1 (System Dynamics Model) A system dynamics (SD) simulation is a variable-based *computational* model for analyzing complex systems containing feedback and feedforward dependencies among variables and rates of change, often with high-dimensionality.

Formally, an SD model consists of a system of discrete-time difference equations with forward or backward differencing. SD models can be purely deterministic or contain stochastic noise as defined by random variables. A complete SD social simulation model consists of causal diagrams explaining the network of dependencies and associated code implementation.

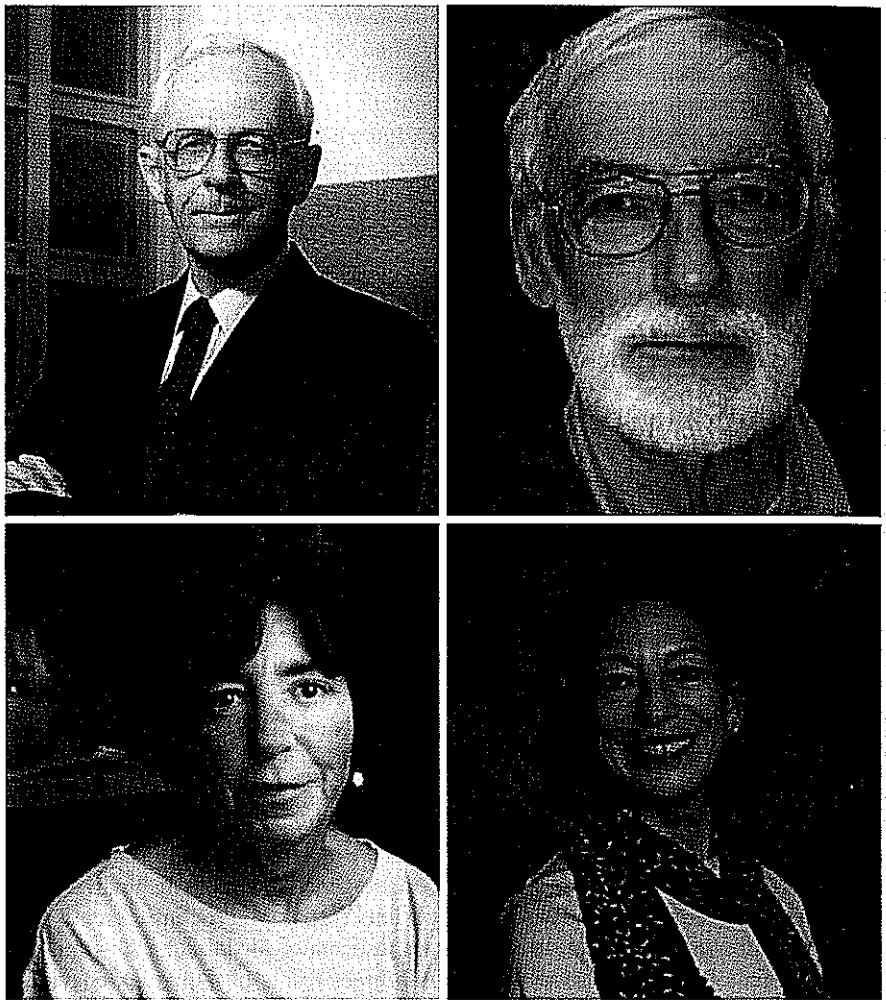

Fig. 9.1 Major pioneers of system dynamics models: Jay Forrester, founder of SD modeling (*upper left*); Dennis Meadows, director of the Club of Rome Project on the Predicament of Mankind, *The Limits to Growth* (*upper right*); Linda Cordell, pioneer in dynamical systems models in archaeology, elected to the National Academy of Sciences in 2005 (*lower left*); Nazli Choucri, MIT pioneer SD modeler of energy, conflict, and state stability dynamics (*lower right*)

Definition 9.2 (Dynamical System Model) A dynamical system (DS) is a variable-based *mathematical* model composed of a set of differential equations or differential and integral equations.

Dynamical system models in social science date to the first pioneering applications to the study of conflict, demographic, and economic dynamics almost a hundred years ago—i.e., they were used in mathematical social science much earlier than computational SD models. Formally, a DS model consists of a system of continuous-time equations. DS models can be purely deterministic or contain stochastic noise defined by random variables. Both SD *and* DS are formal models (computational and mathematical, respectively), and can be purely deterministic or contain stochastic components. The main difference lies in the discrete and continuous time domains, as well as the presence of forward and backward time delays in the former.

9.3.1 Motivation: Research Questions

SD models address research questions in numerous domains of CSS, especially those where the following features are present in a given referent system of interest:

1. Variables and their respective time trajectories are of immediate interest as stocks, sizes, or quantities of some kind. (State variables are later abstracted as levels, as detailed in the next stage of the modeling process.)
2. Causal relations among variables are responsible for observed changes in terms of temporal dependencies; they don't just occur for unknown reasons or through purely random mechanisms. (Change is later abstracted as caused by rates.)
3. Noise can affect resulting trajectories at various points in the causal network. (Noise is later abstracted as probability distributions.)
4. At the macroscopic system level trajectories of change can include stationarity, escalation, dampening, cycling, oscillations, asymptotic behaviors, and other temporal qualitative patterns.
5. Emergent properties of social complexity at the systemic level result from interactions at the level of variables at the lowest causal levels.

9.3.2 Design: Abstracting Conceptual and Formal Models

Given some referent system of interest S, a conceptual model C_S, consisting of a set of state variables and their respective rates of change, is abstracted by a two-stage process rendered through causal loop diagrams and stock and flow diagrams.

9.3.2.1 Causal Loop Diagrams

The first stage in SD abstraction to produce a conceptual model focuses on elementary causal relations called *loops*.

Definition 9.3 (Causal Loop) A causal loop is a feedback relation between a given variable x and its rate of change.

Causal loops are the basic elements of an SD model. In turn, feedback can be positive or negative, depending on whether it promotes or impedes a given variable.

Definition 9.4 (Positive Feedback) A positive feedback loop is a causal relation that increases the value of a variable.

Positive feedback is viewed as a **reinforcement dynamic** in SD terminology, producing an increasing effect: growth, expansion, gains, amplification, increases, improvements, enlargements, proliferation, or escalation, or other increasing patterns in the time trajectory of a variable, depending on the appropriate semantics of the referent system.

Definition 9.5 (Negative Feedback) A negative feedback loop is a causal relation that decreases the value of a variable.

Negative feedback is a said to be a **balancing dynamic** in SD terminology, producing a decreasing effect: fatigue, decline, reduction, loss, diminution, mitigation, depletion, contraction, restraint, decay, or other decreasing patterns in the time trajectory of a variable, again depending on appropriate semantics of the referent system.

Definition 9.6 (Causal Loop Diagram) A causal loop diagram is a graphic abstraction that describes positive and negative feedback in the behavior of a given variable.

Norm adoption by members of a community is an example of an emergent social phenomenon that can be represented by a causal loop diagram. This is useful for understanding how a new norm may be adopted as a social process from an SD perspective, as shown in Fig. 9.2. The figure shows two feedback loops operating simultaneously. The positive feedback loop R, on the right, denotes how social conformity tends to produce new norm adopters by peer pressure as the number of new adopters grows. This is a reinforcement dynamic. The more people conforming to the new norm, the greater the pressure to adopt it, which is abstracted as a positive feedback loop. The feedback loop B, on the left, represents negative reinforcement or "balancing" because the community has finite size, so the number of potential adopters decreases as more community members adopt a new norm. The higher the proportion of conformity with the new norm the lower the number of the non-conformists, so the loop on the left represents negative feedback. Related examples of social norms are fashions, opinions, technological innovations, attitudes, and behavioral patterns, so the norm adoption process has broad applicability across domains of social science.

A similar example is found in the domain of *inter-group conflict*, based on *Richardson's two-group rivalry model of arms race dynamics*, shown in Fig. 9.3. (Although this is sometimes referred to as a two-nation arms race model, Richardson intended it to be a general model for conflict between rival groups of any kind, nations and non-state actors alike, as reflected by his term "deadly quarrels.") In this case the rate of arms acquisition by each group is affected by two opposite dynamics produced by feedback loops. On one hand, there is an escalation dynamic because the rate of arms acquisition is driven by a rival's current (and threatening!) level of arms; the higher that is, the greater the need to catch up by increasing one's own rate. On the other hand, there is a mitigating dynamic driven by the cost of maintaining what one already has, so the higher the level of one's own armaments, the

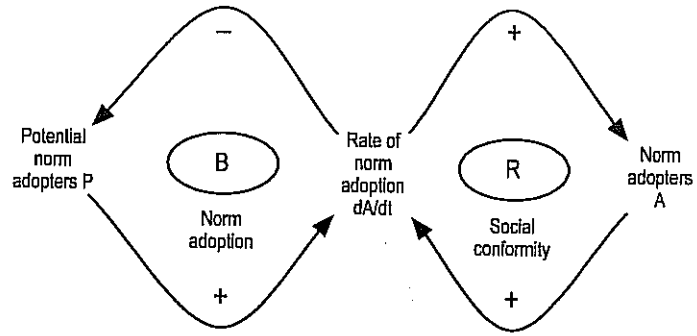

Fig. 9.2 Causal loop diagram for a system dynamics model of norm adoption

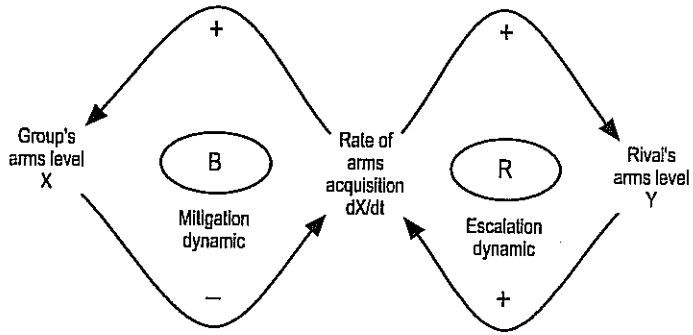

Fig. 9.3 Causal loop diagram for a system dynamics model of inter-group rivalry

greater the economic burden, so the more difficult it is to procure further increases. Today, organization complexity required to support advanced capabilities must be added to direct economic cost. Richardson called this restraining force "fatigue."

A system as a whole is represented by coupled causal loops representing how all elementary causal loops are related to one another. Note that in the last two examples overall system structure is the same, but the signs are not—the balancing signs of the mitigation dynamic are reversed. In the norm diffusion process in Fig. 9.2, the two feedback loops are assumed to be coupled, acting simultaneously. As shown in the diagram, the *rate* of norm adoption is a function of both the *number of potential norm adopters* and the *number of norm adopters*. Potential norm adopters and actual norm adopters are decreased and increased by the adoption rate, respectively. The result is that at different times the two coupled dynamics behave differently. During the early stages of the process, growth in the population of adopters will be greater than in latter stages when fewer non-conformists remain in the community.

In the rivalry process in Fig. 9.3, the two feedback loops are also assumed to be coupled, so they operate simultaneously. The *rate* of arms acquisition is a function of both the *rival's arms level* and the *group's (own) arms level*. The escalation dynamic on the right is a self-reinforcing drive (positive feedback). The mitigation dynamic

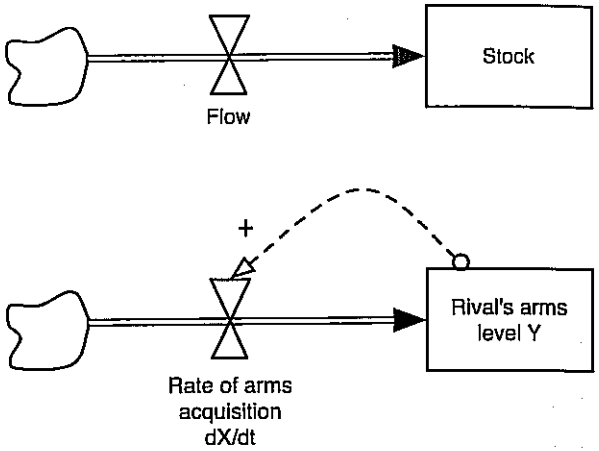

Fig. 9.4 SD stock and flow diagram for representing variables (stocks represented as *rectangles*) and rates of change (flow represented as *valves*)

on the left is a balancing drive (negative feedback). However, unlike the previous example, this case assumes *two different kinds couplings*, both acting on the rates of arms acquisitions:

1. Feedback couplings: positive and negative feedback processes are coupled, as in the norm emergence example.
2. Actor couplings: the two rivals are coupled through strategic interaction, in a game-theoretic sense, since the outcome for each (arms levels) is determined not only by what one decides, but also by what the rival decides.

These two coupled dynamics in the rivalry process in this case also behave differently at different times, depending on which dynamic drive prevails.

In sum, causal loop diagrams can contribute to building a conceptual SD model from a qualitative perspective by abstracting positive and negative feedback loops corresponding to reinforcing/escalating and dampening/mitigating drives, respectively. However, more is needed to build a sufficiently complete conceptual model of a referent system that can be computationally implemented in code.

9.3.2.2 Stock and Flow Diagrams

The second stage of abstraction in SD model development is to provide a more quantitative way of representing system structure and dynamics using stock and flow diagrams, as shown in Fig. 9.4. In this second kind of SD diagram, variables become stocks (rectangles) and rates become flow valves (bow ties). Unlike a feedback loop diagram, a stock and flow diagram can be directly translated into code.

The top of Fig. 9.4 shows a generic stock and flow diagram with its basic notation, where the source on the left represents a variable with realization determined by the flow valve that controls the stock or level on the right. The bottom of the figure uses the same notation applied to the case of the reinforcement loop or escalation dynamic of a conflict process (right portion of Fig. 9.3), where a group's rate of acquisition in military capabilities is determined by the level of its rival. The fully coupled conflict system is shown in Fig. 9.5.

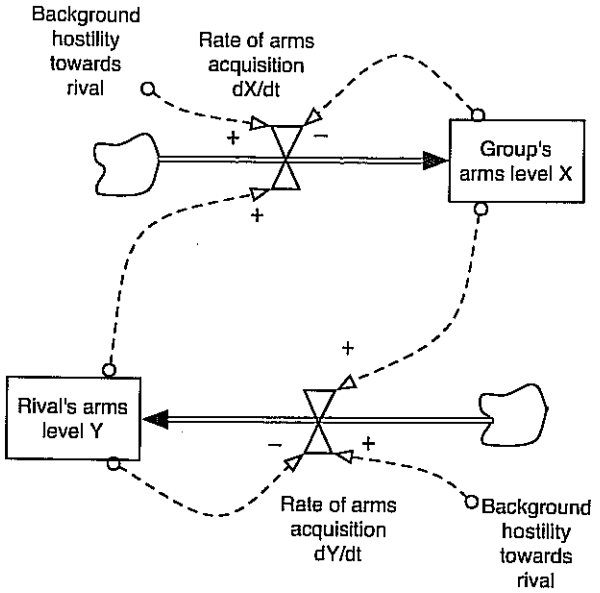

Fig. 9.5 Stock and flow diagram for a system dynamics model of a two-group rivalry interaction

Figure 9.5 specifies how rival actors and feedback loops are mutually dependent on each other, to formalize the concept of strategic interaction. The figure uses the same basic stock and flow components as in Fig. 9.4, with the added element of background hostility acting as a parameter that also affects the rate of change, so now the dynamic process of each rival is driven by three factors:

1. The rival's current arms level (representing positive feedback, escalation force)
2. The group's own arms level (negative feedback, mitigation force) and
3. Background hostility acting as a constant background force, which captures the idea that a group would acquire some minimal military capabilities as insurance, regardless of a rival's arms level.

Diagrams such as these—usually involving many more stocks/variables, flows/rates, and parameters—are used in SD methodology for representing a conceptual model of a given referent system. Noise, stochastic shocks, and other elements are also added as necessary.

The main result of the design stage in system dynamics is a conceptual model of the referent social system specified by a set of equations. For example, in the conflict model, the following system of equations in continuous time specifies the rivalry dynamics:

$$\frac{dX}{dt} = aY - bX + g \tag{9.1}$$

$$\frac{dY}{dt} = \alpha X - \beta Y + h, \tag{9.2}$$

where a and α are reaction coefficients, b and β are mitigation coefficients, g and h are hostility coefficients, and X and Y are levels of armaments. The following system of equations is in discrete time:

$$X(t+1) = aY(t) - bX(t) + g \tag{9.3}$$

$$Y(t+1) = \alpha X(t) - \beta Y(t) + h. \tag{9.4}$$

In this case, the system of equations can be analyzed to obtain closed form solutions, since the system is simple. Solutions to these systems of equations yield time trajectories containing exponential terms, which can be easily verified. In most cases this is not possible, which is why simulation is required.

9.3.3 Implementation: System Dynamics Software

Given a sufficiently complete conceptual model of a referent system, the next stage in SD methodology consists of implementing the model in code using a simulation system. The key milestone activity in the implementation stage is marked by the transition from mathematical equations in the conceptual model to code in the simulation model.

The current, most utilized simulation system for implementing SD models is called VENSIM, which is the current successor to earlier DYNAMO and STELLA simulation systems software. Vensim PLE is an education version that is made available free of charge. The classic textbook by John D. Sterman, *Business Dynamics*, includes a CD (for PC and Macintosh) containing simulation software and models, including ithink, Powersim, and Vensim software. A major advantage of systems such as these is their close association with the SD community, specifically the System Dynamics Society. The Vensim website has numerous resources for beginning and advanced users, including tutorials and other helpful materials.

Figure 9.6 shows a screenshot of the Vensim system while implementing a conceptual stock and flow model of a simple customer base in a company. While Dynamo was a programing language that required writing code, Vensim can be used by selecting facilities for defining variables, equations, and other components by clicking options, using drop-down menus, and other features of the user interface.

Another option for developing SD social simulation is to implement the conceptual model in simulation systems such as Netlogo or Repast. Although these simulation systems were not originally designed to run SD models, they do have such facilities in addition to the agent-based models for which they were originally designed. For example, Netlogo has demonstration SD models for exponential growth,

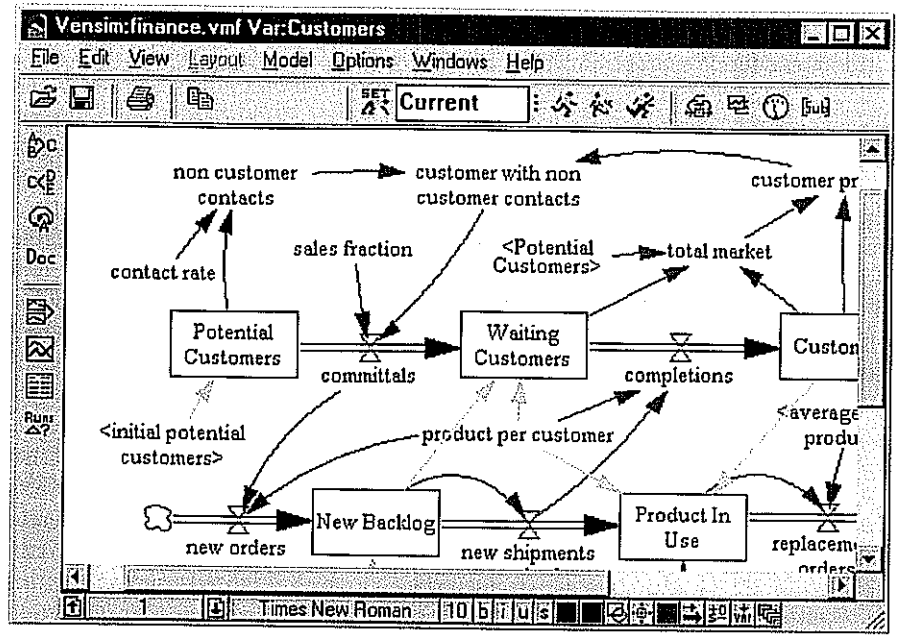

Fig. 9.6 Screenshot while implementing an SD social simulation using the Vensim system

logistic growth, prey-predator (wolf-sheep) dynamics (based on the classic Lotka-Volterra model), as well as other effective examples.

9.3.4 Verification

Recall the difference between verification and validation: the former is about ensuring a model is running the way it is supposed to, as guided by the conceptual model and any other simulation design specifications; the latter is about ensuring that the simulation model is a good representation of the referent system.

Once an SD social simulation model has been implemented, the next step involves verification procedures. Systems such as Vensim provide a number of facilities for verifying a model, such as checking that the right units are specified, rates are using the proper dependencies, and similar steps to ensure that the model is running the way it was intended by the conceptual model. Since an SD conceptual model, complete with stock and flow diagrams, uses the iconic metaphor of levels and flow valves, verifying an SD implementation essentially means checking that all "the plumbing" is working as it should according to the most minute details in the blueprints (stock and flow diagrams). Each element must be checked for accurate implementation, as well as every rate, feature, and connection. Facilities provided

by whatever simulation system is chosen should be used in the context of the verification procedures examined in Chap. 8.

9.3.5 Validation

Validating an SD social simulation model that has been verified is accomplished from two main perspectives. **Structure validity** refers to internal features of the model, including all assumptions, relevant variables and their units, and the system of equations in all its component stocks and flows. The following are recommended tests of structure validity for SD models:

Empirical tests of validation: This is aimed at validating the specification of equations used in the model as well as parameter values being used. For example, in the case of the conflict model discussed earlier, this part of the validation process would focus on parameters such as the equation's coefficients being assumed, as well as constants, such as background hostility that affects armament rates. The equations themselves require validation, since different specifications will yield different results. The classic rivalry model assumes additive and symmetrical armament levels, which is an assumption that requires validation through using empirical tests. It is also assumed that reaction coefficients and hostility parameters are constant. These all add up to an overall assumption of structural stationarity, in the sense that all equations specified do not undergo significant change over time—i.e., the standard model assumes that the basic clockwork mechanism does not change as history evolves, which may or may not be a valid assumption.

Theoretical tests of validation: Model assumptions should also be confirmed by the extant theories being used, since even the simplest SD model assumes theoretical mechanisms that justify its causal structure. This is a broader perspective than empirical tests of structural validity, since it is based on fundamental causal arguments that are difficult if not impossible to quantify. For example, in the case of the conflict model, the overall structure is grounded on Richardson's theory of how rivalry between two groups is explained. The fundamental theory is based on three factors or dynamics driving the conflict process: escalation forces driven by positive feedback from a rival's stock of weapons; mitigation forces driven by fatigue and negative feedback from one's own stockpile of armaments; and some background constant force generated by hostility over disagreements and insecurity. Is this theory valid? Are there other factors as important or even more significant than these? The theory also assumes perfect symmetry between rivals; both make arms procurement decisions in the same way. Is it possible that the rivals in question decide with different goals, such as one trying to "catch up" with the other, so it reacts to the *gap* between its own level and the rival's [i.e., $dX/dt \propto (X - Y)$], not simply to the rival's level ($dX/dt \propto Y$ as in Eq. (9.1))?

As with any other kind of social simulation model, tests of structural validity for SD models are complex and require considerable attention. The empirical literature is of great value in navigating through these procedures.

By contrast, **behavior validity** concerns the actual results of simulation runs, primarily in terms of qualitative and quantitative features such as patterns of growth, decay, and oscillation, among others. Many of these procedures involve various forms of time-series analysis and extensions. Some of these were mentioned during the general methodological discussion in the previous chapter, including analyzing trends, comparing periodicities by means of autocorrelation functions, comparing distribution moments, and computing global statistics such as the discrepancy coefficient between simulated and observed time-series data (Barlas 1996: 207–208).

9.3.6 Analysis

The main goal of simulation research in CSS is to obtain qualitative and quantitative results to better understand the referent system. The previous forms of qualitative and quantitative analysis are primarily procedural, for purposes of gaining confidence in the veracity of a model by conducting verification and validation procedures. Obviously, the main goal of developing an SD social simulation—the reason for going through all the trouble—is to analyze it in substantive ways. Formal analysis, asking what-if questions, and scenario analysis constitute major forms of analyzing SD social simulations.

Formal analysis of an SD model yields results, such as *time trajectories* for all level variables (stocks), *phase portraits* in parameter spaces, *sensitivity analysis*, *comparative statics*, and similar sets of results in dynamical systems analysis. For example, the conflict model results from formal analysis would show the time trajectories of levels of armaments in the evolution of conflict between groups, phase portraits of trajectories as a function of parameter combinations, and similar qualitative and quantitative results. Results from formal analysis can reveal properties such as orbits, singularities, asymptotes, attractors, gradient fields, periodicities, chaos, bifurcations, ergodicities (equality between time averages and space averages), phase transitions, stability properties, and other significant dynamic features of social complexity that are typically not apparent from the model structure.

Asking **what-if questions** is another major approach to analyzing SD social simulations. For example, in the conflict model we may ask what happens when the hostility of one group versus its rival is some multiple of the other's hostility. Or, what happens when reaction coefficients differ significantly across the two groups? What-if questions can also extend to analysis of an SD model with alternative specifications of equations to explore what happens when rates of change are governed by different dynamics. For example, as was suggested earlier, in the conflict model we may wish to have a rival responding to the gap $(Y - X)$ in armament levels, as opposed to the original assumption of responding to just level Y.

A more comprehensive form of analysis used with SD social simulations is **scenario analysis**, which typically involves a suite of related questions defining a given scenario, rather than analyzing one question at a time. For example, in the conflict model we may wish to examine a scenario in which reaction coefficients are relatively small, mitigation coefficients are several times larger than reaction coefficients, and hostility coefficients are weak. Intuitively, such a scenario should lead