the marketing of a new product) commences at one clear moment in time. These are issues that one needs to watch out for; there is no general solution that always applies.

Note

1. A number of subsequent papers by different authors have used such tags to explore the emergence of differences between groups of agents (e.g. Edmonds, 2006; Hales, 2000, 2002; Riolo, Cohen, & Axelrod, 2001).

3. USING AGENT-BASED MODELS IN SOCIAL SCIENCE RESEARCH

Over the past decade, agent-based modeling has developed a more or less standardized research process, consisting of a sequence of steps. Like most social science methods, this is an idealization of the procedures actually carried out, and in practice, several of the steps occur in parallel and the whole process is performed iteratively as ideas are refined and developed. Nevertheless, it is useful to have these steps made explicit as a guide to the conduct of agent-based modeling research.

At an early stage in the research, it is essential to narrow down the general research topic to some specific research question. A *research question* is something that the work should have a realistic chance of answering. If the question is too vague or too general, it will not be much use, and the research will be disappointing as it will not be able to provide the hoped-for answers. It is always better to err on the side of specificity: Be too focused rather than too ambitious. It is sometimes helpful to think of defining the research question as like stripping away the layers of an onion, from the general area of investigation, through the particular topic, to a question that could be answered in no more than a brief statement of what you have discovered.

As we have noted above, the usual kind of research question that agent-based models are used to study are ones where regularities at the societal or macro level have been observed, and the issue is how these may be explained. Economists often call these regularities *stylized facts* (Kaldor, 1961). For example, the Schelling model described in Chapter 1 starts with the observation that neighborhoods are ethnically segregated and seeks to explain this through modeling individual household decisions. The electricity market models also described in Chapter 1 aim to explain (and predict) patterns of electricity supply and market pricing in terms of the motivations of suppliers.

After having specified the research question clearly and identified the macro-level regularities that are to be explained, the next step is to specify the agents that are to be involved in the model. They may be all of one type, or there may be different types. For example, while the models of opinion dynamics reviewed in Chapter 1 involve only one type of agent, the individuals whose opinion changes are being simulated, some of the industrial districts models mentioned in Chapter 1 involved several distinct types of firms. For each type of agent, one needs to lay out the agent's behavior in different circumstances, often as a set of condition-action rules (see Section 2.1.2). It is helpful to do this in the form of two lists: one that shows all the different ways in which the environment (including other agents) can affect the agent, and one showing all the ways in which the agent can affect the environment (again, including other agents). Then, one can write down the conditions when the agent has to react to environmental changes, and the conditions when the agent will need to act on the environment. These lists can then be refined to create agent rules that show how agents need to act and react to environment stimuli.

At this stage, one will have a good idea of the types of agents and their behaviors that are needed in the model. It will also be necessary to consider what form the environment should take (for instance, does it need to be spatial, with agents having a definite location, or should the agents be linked in a network?) and what features of the model need to be displayed in order to show that it is reproducing the macro-level regularities as hoped. Once all this has been thought through, one can start to design and develop the program code that will form the simulation.

After the model has been constructed, one begins the long process of checking that it is correct. Informally, this is called debugging; more formally, it is *verification*. Verification is the task of ensuring that a model satisfies the specification of what it is intended to do. It is quite different from *validation*, which is checking that the model is a good model of the phenomenon being simulated. One can have a simulation that satisfies the verification criterion, because it runs as it is supposed to do, but if the specification is a poor description of the target in the social world, it is not a valid model.

Following successful verification, one can embark on validation. The primary criterion of validation is whether the model shows the macro-level regularities that the research is seeking to explain. If it does, this begins to be evidence that the interactions and behaviors programmed into the agents explain why the regularities appear. However, one must guard against alternative explanations. There may be other, equally or more plausible agent behaviors that lead to the same macro-level regularities. Therefore, one needs to engage in a *sensitivity analysis* to see whether, when model parameters are changed, the outcomes alter, too. It is also important to

consider whether quite different agent behaviors could lead to the same results, in particular whether a simpler model leads to the same conclusions (if it does, the simpler model should normally be preferred to the more complicated one, using the principle that simple explanations are better than complicated ones if both are equally good at explaining).

Having thus explored the macro behavior of the model, it is then desirable to compare the output of the model with empirical data from the social world. As we shall see later, such comparisons between model outputs and data are not easy to carry out and often do not lead to the clear answers that one might expect. Most models are stochastic, that is, involve random processes, so one does not know whether any difference between the model output and observed data is due to random chance or a bad model. There are also often considerable difficulties in collecting valid and reliable data, especially the data observed over long periods of time that one needs to compare with model outputs.

Finally, one can draw some conclusions, hopefully answering the research question that started the process. In addition, if one has confidence in the model, one can experiment with it, perhaps to identify regularities that had been previously unsuspected.

3.1 An Example of Developing an Agent-Based Model

In this section, the process of developing a simple agent-based model will be described using a simulation of ''collectivities'' (Gilbert, 2006) as an example. In Chapter 4, we shall see how this model could be programmed.

A number of related social phenomena are hard to model, or even to describe, because their boundaries are fluid, the people involved are constantly changing, and there is no single characteristic shared by all those involved. Examples include the following:

- Youth subcultures, such as ''punks'' (Widdicombe & Wooffitt, 1990) or ''goths'' (Hodkinson, 2002)
- Scientific research areas or specialties (Gilbert, 1997)
- Art movements such as the Pre-Raphaelites or the Vorticists (Mulkay & Turner, 1971)
- Neighborhoods, such as Notting Hill in London or the Bronx in New York (O'Sullivan & Macgill, 2005)
- Members of armed revolutionary or terrorist movements (Goolsby, 2006)
- Industrial sectors such as biotechnology (Ahrweiler, Pyka, & Gilbert, 2004)

Although one can easily point to familiar examples, and although they are very common and easily identified, it is difficult to put one's intuitions about them on a firmer footing. For a start, there is no commonly accepted word with which to name the phenomenon. The terms *subculture, area, neighborhood, specialty,* and *movement* are used for particular types, but none of these words is appropriate for describing all of them. A closely related concept is "figuration" (Elias, 1939/1969), although strictly this should be applied only to individuals, not to organizations or other types of actors. In this section, we use the term *collectivity* as the generic term, for lack of a better one. Note that the units making up the collectivity may be people (as in most of the examples above) or organizations (e.g., biotechnology firms).

A second barrier to gaining a better understanding of collectivities is that, by definition, there is no definite boundary around them. This means that it is impossible to count their members and therefore to engage in the more common kinds of quantitative analysis of their development over time, their incidence, and so on.

Third, the way in which collectivities arise from the actions of their members is not easily understood. It is the purpose of the model to be developed here to suggest how some plausible assumptions about individual action (*micro foundations*) could yield the collectivities that are observable at a macro level.

3.1.1 Macro-Level Regularities

In all collectivities, the following seem to hold, to a greater or lesser extent:

- Although instances of collectivities are usually easily named and described at the aggregate level, precise definitions can prove to be rather slippery and open to negotiation or argument (e.g., there are many slightly different areas that can be described as Notting Hill, from the official local government area to the locality within which the film of the same name was shot).

- There is no accepted consensual definition that can be used to sort those who are "in" from those who are "out" (or members from nonmembers). For example, whereas some might think that a person is a "punk" because of the way that he or she dresses, this assignment might be contested by others (including the person him- or herself) by pointing to the person's beliefs, behavior, or acquaintances, all of which could alternatively be relevant for making a decision on membership. In particular, there is no one observable feature that all those who are "in" and none of those who are "out" possess. Collectivities are not, for example, formal organizations, where being an employee with a written or verbal contract distinguishes

those who are members; political parties, where, at a minimum, a formal declaration of support is required and defines membership; or social classes, where externally specified objective criteria are used to sort people (typically one's occupation).

- Nevertheless, many of the members will share characteristics in common (e.g., the scientists in a research area may have similar education, have carried out similar previous research, and be known to each other, even if there is no technique, theory, or object of research with which all of those without exception in the research area are involved).

- Membership of the collectivity entails possessing some related knowledge (e.g., the science of the specialty, or whatever is accepted as ''cool'' in a youth culture, or the local geography of Notting Hill). However, no member possesses all the knowledge: Knowledge is socially distributed.

- The features that are thought to be relevant to the collectivity change. For example, researchers do not continue to work on exactly the same problems indefinitely; once they have solved some, they move on to new ones, but still within the same research area. Most political movements change their manifestos over time to reflect their current thinking and the social problems that they see around them, although they remain the same movements, with many of the same adherents. Youth cultures are constantly changing the items that are considered to be in fashion.

- Some of the people involved are widely considered (e.g., by the others) as being more central, more influential, of greater status, or as leaders as compared with others. For example, some scientists are considered to be more eminent than others, some members of subcultures are more ''cool'' than the rest, and so on.

3.1.2 Micro-Level Behavior

One of the features common to collectivities mentioned in the previous section is that the actors (that is, the people or organizations that make up the collectivity) have some special knowledge or belief (e.g., for scientists, knowledge about their research area; for youth subcultures, knowledge about what is currently fashionable). Even though this knowledge is socially distributed among the members of the collectivity, so that not every member has the same knowledge, possession of it is often a major feature of the collectivity (Bourdieu, 1986). In the model, we assume that all individuals, members and nonmembers, have some knowledge, but what this knowledge is varies both between actors and over time. We use this knowledge to locate the actors: The position of the actor at a moment in time in an abstract knowledge space is a function of the knowledge that he or she possesses at that time.

A second assumption is that some actors are of higher status than others and that all actors are motivated to try to gain status by imitating high-status actors (by copying their knowledge). For example, in a collectivity driven by fashion, all actors will want to be as fashionable as they can, which means adopting the clothing styles, musical tastes, or whatever of those whom they perceive to be of the highest status (Simmel, 1907). However, status is also a function of rarity: An actor cannot remain of high status if there are many other actors with very similar knowledge. For example, a fashion icon must always be ahead of the hoi polloi; a scientist will be heavily cited only if his or her research is distinctive; a revolutionary will earn the respect of colleagues only if he or she stands out in comparison with the "foot soldiers."

Third, we assume that the highest status actors want to preserve this status, which they cannot do if they start to be crowded out by followers who have been attracted to them. In this situation, we assume that high-status actors are motivated to make innovations, that is, to search out nearby locations in knowledge space where there are not yet crowds.

There are thus two countervailing tendencies for actors—on one hand, they want to get close to the action; on the other, they want to be exclusive and can do so by changing the locations that represent the heights of status. As we shall see, working out this tension yields patterns at the macro level that are typical of collectivities.

3.1.3 Designing a Model

Related Models

There are several generic models that deal with similar issues:

1. *Boid models* (Reynolds, 1987) have agents that try to maintain a desired distance away from all other agents and thus appear to move with coordinated motion. Agents have three steering behaviors: separation, to avoid nearby agents; alignment, to move in the same direction as the average of nearby agents; and cohesion, to move toward the average position of nearby agents. The effect is that agents move as in a flock of sheep or a school of fish. These models illustrate the effect of having agents carrying out actions that are in "tension": The separation behavior is in tension with the cohesion behavior, for instance. However, there are no notions of seeking status or innovation in these models.

2. *Innovation models* (e.g., Ahrweiler & Gilbert, 1998) have agents that are able to learn and act according to their current knowledge. Agents also exchange knowledge and create new knowledge. However, there is no

specific idea of collectivity in these models. The set of agents involved in innovation is predetermined.

3. *The minority game* (Slanina, 2000) is one example from a large literature. This model, also called the El Farol Bar model, has agents who wish to go to the bar, but only when a minority of the other agents also choose to go there. The agents make a decision based on their own past experience of the number they previously encountered at the bar. Each agent has a number of strategies that he or she uses in combination with his or her memory of the outcome of recent trips to the bar to make a decision on whether to visit the bar at the current time step. The strategies are scored according to their success (whether, when the agent arrives at the bar, it is overcrowded or not), and unsuccessful strategies are dropped. Over time, a dynamic equilibrium can be established, with the number of agents at the bar matching the threshold that agents use to judge that there are too many agents there. This model has some features of the problem addressed herein, but there is no representation of a collectivity.

The Model

The collectivities model consists of a surface over which agents are able to move. The surface is a *toroid* with each point representing one particular body of knowledge or set of beliefs. The agents thus move, not in a representation of physical space, but rather in "knowledge space." Although it may be oversimplifying to represent a knowledge space in two dimensions (more exactly, on the surface of a toroid), it makes for easier visualization.

An agent's movement in the knowledge space represents its change in knowledge. Thus, if an agent imitates another agent, it would be viewed as moving toward that agent in the knowledge space, whereas if it innovates and discovers knowledge that other agents do not have, it would be viewed as moving away from other agents into previously empty areas of the space.

Agents are initially distributed at random on the surface. Agents have no memory of their own or other agents' previous positions. Each agent does the following:

1. It counts how many other agents there are in its immediate neighborhood.
2. If the number of agents is above a threshold, it turns to the direction opposite to the average direction of travel of other nearby agents and then moves a random distance.

3. If the number of agents is equal to or below the threshold, it looks around the locality to find a relatively full area and then moves a random distance from its present location in the direction of the center of that area.

Each agent acts asynchronously, repeating this sequence of actions indefinitely. There are four parameters required by this algorithm (see Figure 3.1):

1. The radius of the circular area surrounding an agent within which the number of agents is counted to determine whether the agent is "crowded" or "lonely" (*local-radius*)
2. The threshold number of agents below which the agent is "lonely" and above which the agent is "crowded" (*threshold*)
3. The radius of the circular area surrounding an agent in which the agent, if lonely, counts the number of agents to find where there is a maximum or, if crowded, finds the average direction of agent movement in order to determine the direction in which it is to move (*visible-radius*)

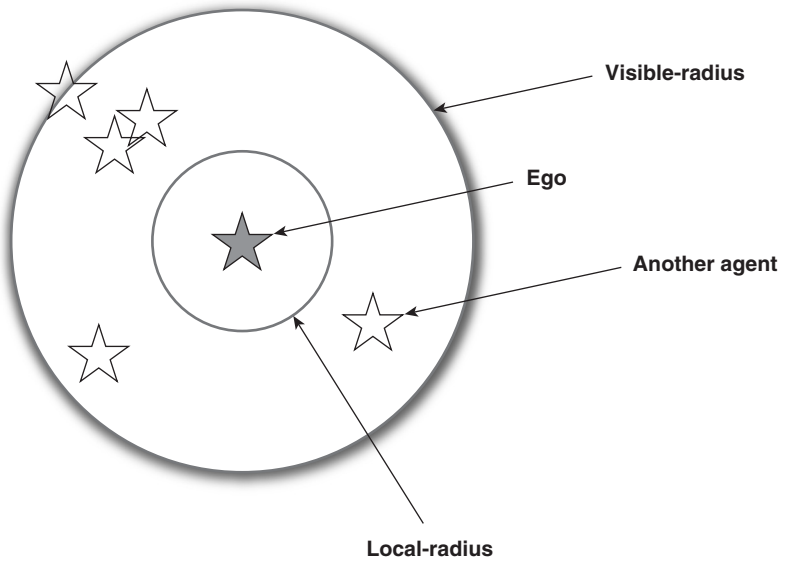

Figure 3.1 Parameters for the Collectivities Model

4. The distance that an agent moves; the distance is chosen randomly from a uniform distribution with this parameter as the maximum (*speed*)

3.1.4 Verification

The verification and validation of the collectivities model will be discussed in Chapter 4, after we have described how to program it. First, we shall consider some general issues that arise in verifying agent-based models.

3.2 Verification: Getting Rid of the Bugs

You should assume that, no matter how carefully you have designed and built your simulation, it will contain bugs (code that does something different from what you wanted and expected). When you first run a new simulation model, it is likely that it will have many bugs, most of them easily observable because the simulation gives anomalous results or crashes. More worrying, it is quite likely that even when you have worked on the code to remove the obvious bugs, some will still lurk to catch you unaware. As a rule of thumb, think of the number of bugs remaining as following a negative exponential—the number decreases rapidly at first, but then levels off and never hits zero. Even published simulations sometimes seem to suffer from bugs and misinterpretations (for examples, see Edmonds & Hales, 2003; Galan & Izquierdo, 2005; Rouchier, 2003).

There are techniques for reducing the chances of including bugs and for making finding them easier (Ramanath & Gilbert, 2004). Here are some:

- *Code Elegantly*. When you are writing the simulation program, do it carefully and steadily; don't rush to get the code working and take shortcuts. Any time saved will be lost in extra time needed for debugging. Using an object-oriented language and using variable names that are meaningful in the context of your model will help.

- *Include Lots of Output and Diagnostics*. It will be hard to find bugs in sections of code that give no output to show what is happening as the program runs. Don't be satisfied with only displaying the results of the simulation; at least during the debugging phase, you will need to display intermediate values also. Some care will be needed to decide what to display so that you get good diagnostics, but are not so overwhelmed that you cannot find the symptoms of the bugs because of the amount of other output in which they are buried.

• *Observe the Simulation, Step by Step*. Run the code one line or one function at a time, observing how the values of variables, parameters, and attributes change and checking that they alter in the expected way. Although this can be slow and tedious, it does help to ensure that the code is doing what was intended, at least for the runs that are observed. Often, programming environments provide features to make stepping through code easier to manage.

• *Add Assertions*. If you know that variables must take some values and not others, check for valid values as the simulation runs, and display a warning if the value is out of range (such checks are known as assertions). For example, if two agents cannot occupy the same spatial location, at each time step check and report if this requirement is violated.

• *Add a Debugging Switch*. You may worry that all the code needed to identify bugs, such as assertions and diagnostics, will slow down the simulation unacceptably. Include a global variable in your program that can be set to a debugging level: from none to maximum. Precede each debugging statement with a test of this variable, to see whether the statement should be run at the current debugging level (it is possible in some languages to achieve the same effect with conditional compilation).

• *Add Comments and Keep Them Up to Date*. All programming languages allow you to insert comments—text for programmers to read that is not executed as program code. Use this feature to add comments to every function, procedure, method, and object. The comments should describe what the following block of code does and how it does it, but at the conceptual level, not at the implementation level (that is, do not paraphrase the program code, but state what the code is intended to achieve). As a rule of thumb, there should be about one third as many lines of comment as there are lines of code. Comments can easily become obsolete, describing the program as it used to be, rather than as it is. Reserve some time to update the comments at regular intervals. In writing comments, assume that the reader is someone who can program as well as you can, but who knows little or nothing about your model (after a couple of months away from the program, this may be a good description of you, so do not avoid writing comments with the excuse that no one else will see your code).

• *Use Unit Testing*. Unit testing is an increasingly popular software engineering technique for reducing bugs (Link & Fröhlich, 2003). It consists of writing some test code to exercise the program at the same time as you write the code itself. The idea is to develop the program in small, relatively self-contained pieces, or units. A test "harness" is created that will supply the unit with a sequence of inputs and check the results against a list of

expected outputs. The test harness then automatically runs through each input, checking that the expected output is, in fact, generated. Once the unit has passed all of its tests, you can move on to writing the next unit. This may involve making some changes to the first unit, which must then be put through its test sequence again to ensure that the changes have not introduced any new bugs. When many units have been written, the test harness is used to automate performing all the tests on all the units, thus giving some guarantee that bugs have not inadvertently crept in as a result of developing the code. As the program develops, additional tests should be written to verify assemblies of several units and the interaction between them.

- *Test With Parameter Values for Known Scenarios.* If there are any scenarios for which the parameters and the output are known with some degree of certainty, test that the model reproduces the expected behavior. This is the test that most people carry out first of all on a model, but it is a rather weak test and, by itself, will not give much confidence that the simulation is free of bugs.

- *Use Corner Testing*. Test the model with parameter values that are at the extremes of what is possible and ensure that the outputs are reasonable (the name comes from the idea that such parameter values mark the corners of a parameter space enclosing all possible parameter values). For example, test to see what happens when your simulation is run without any agents and when it is run with the maximum number that your model allows.

3.3 Validation

Once one has developed an agent-based model, it seems obvious that one needs to check its validity, that is, whether it is, in fact, a good model of what it purports to represent. However, both the theory and practice of validation are more complicated, and more controversial, than one might at first expect. The issues are related to the various objectives aimed at by modelers, which imply different criteria for validation, and the sheer difficulty of acquiring suitable social science data in sufficient quantity to allow systematic validation (Troitzsch, 2004). We begin by considering the conceptual issues before discussing some techniques for carrying out validation.

Agent-based models can be directed primarily at formalizing a theory (for example, Schelling's residential segregation model; see Chapter 1), in which case the model is likely to be pitched at a very abstract level; or they can be aimed at describing a wide class of social phenomena, such as the development of industrial districts or the behavior of consumers; or they can be intended to provide a very specific model of a particular social situation, such as some of the models of electricity markets mentioned in

Chapter 1, where the precise characteristics of one market, including the location of power plants and pattern of consumer demand, are relevant. These types of agent-based models require rather different approaches to validation (Boero & Squazzoni, 2005).

3.3.1 Abstract Models

The aim of abstract models is to demonstrate some basic social process that may lie behind many areas of social life. A good example is Epstein and Axtell's pioneering book on *Growing Artificial Societies* (Epstein & Axtell, 1996), which presents a series of successively more complex models of the economics of an artificial society. Another example is the model of collectivities introduced earlier in this chapter. With these models, there is no intention to model any particular empirical case, and for some models, it may be difficult to find any close connection with observable data at all. For example, Schelling's model is usually built on a toroidal regular grid with agents dichotomized into two classes (e.g., red and green). These characteristics of the model are plainly not intended as empirical descriptions of any real city or real households. How then might such models be validated?

The answer is to see such models as part of the process of development of theory, and to apply to them the criteria normally applied to evaluating theory. That is, abstract models need to yield patterns at the macro level that are expected and interpretable; to be based on plausible micro-level agent behavioral rules; and, most important, to be capable of generating further, more specific or "middle range" theories (Merton, 1968). It is these middle range theories and the models based on them that may be capable of validation against empirical data. If an abstract model has been created using a deductive strategy, there will already be some hypotheses about the agent's behavior and about the macro-level patterns that are to be expected. The first validation test is therefore to assess whether the model does indeed generate the expected macro-level patterns. A more thorough test would be to see what happens when parameters of the model are systematically varied (see Section 3.4.1, on sensitivity analysis). One would hope that either the macro-level patterns persist unchanged with variation in parameters, or, if they do change, the changes can be interpreted. For example, in Schelling's model, one can alter the tolerance level of the agents. At low values of tolerance, households rarely find a spot where they are happy and the simulation takes a long time to reach a steady state, if it ever does. At sufficiently high values of tolerance, households are satisfied whatever the color of their neighbors and the initial random distribution hardly changes.

Once these basic tests have been passed, one can evaluate whether the model can be used to inform theories about specific social phenomena, and

then those theories can be tested. For example, the Schelling model, although developed for ethnic segregation, is more general and abstract than this implies. It could be applied to any characteristic of actors. When used to explain ethnic segregation, however, one needs to start developing the theory to include the other factors that are of undoubted importance in location decisions in urban areas, including affordability and availability of the housing stock, the presence of more than two ethnic groups and people who belong to none or more than one, and the functional form of ethnic attitudes. (For instance, Bruch and Mare, 2006, suggest that the segregation effect in the Schelling model depends on the agents having a dichotomous attitude of being either happy or unhappy, and that clustering does not result if agents have a smoothly continuous attitude ranging from very unhappy to very happy.)

3.3.2 Middle Range Models

Models such as those mentioned in Chapter 1 that simulate consumer behavior, industrial districts, or innovation networks are intended as "middle range" simulations: They aim to describe the characteristics of a particular social phenomenon, but in a sufficiently general way that their conclusions can be applied widely to, for example, most industrial districts rather than just one.

The generic nature of such models means that it is not usually possible to compare their behavior exactly with any particular observable instance. Instead, one expects to be satisfied with qualitative resemblances. This means that the dynamics of the model should be similar to the observed dynamics and that the results of the simulation should reveal the same or similar "statistical signatures" as observed in the real world; that is, the distributions of outcomes should be similar in shape (Moss, 2002).

For example, the firms that one finds in innovation networks have collaborative links with other firms in the same industrial sector. If one counts the number of partners of each firm and plots the log of the number of partners against the log of the number of firms with that many partners, the graph is approximately a straight line with constant slope (e.g., Powell, White, Koput, & Owen-Smith, 2005, Figure 3). A linear relationship between logged variables is the statistical signature of a *power law,* and it is characteristic of many social networks, from utility power networks to the World Wide Web (Barabási, 2003). We would expect that a simulation of an innovation network would also show a power law distribution of inter-firm links with a similar slope.

An example of a middle range model is Malerba, Nelson, Orsenigo, and Winter's (2001) work on the computer industry. They describe their model

as "history-friendly," by which they mean that while the model does not reproduce the exact history of the computer industry, it does

> capture in a stylized and simplified way the focal points of an appreciative theory about the determinants of the evolution of the computer industry. It is able to replicate the main events of the industry history with a parameter setting that is coherent with basic theoretical assumptions.

They also note that "changes in the standard set of parameters actually lead to different results, 'alternative histories' that are consistent with the fundamental causal factors of the observed stylized facts."

3.3.3 Facsimile Models

Facsimile models are intended to provide a reproduction of some specific target phenomenon as exactly as possible, often with the intention of using it to make a prediction of the target's future state, or to predict what will happen if some policy or regulation is changed. For example, a business may be interested in finding the consequences for their inventory level of reducing the interval between sending out restocking orders. It is likely to require a model that precisely represents all their suppliers, the goods each supplies, and the unit quantities of those goods in order to be able to make reasonable predictions. Another, very different example is the work by Dean et al. (1999) on the Anasazi Indians in the southwestern United States. These people began maize cultivation in the Long House Valley in about 1800 BC, but abandoned the area 3,000 years later. Dean et al.'s model aimed to *retrodict* the patterns of settlement in the valley and match this against the archaeological record, household by household.

If such exact matches can be obtained, they would be very useful, not only as a powerful confirmation of the theory on which the model is based, but also for making plausible predictions. However, there are reasons for believing that simulations that exactly match observations of specific phenomena are likely to be rare and confined to rather special circumstances. Most social simulations contain some element of randomness. For example, the agents may have initial characteristics that are assigned from a random distribution. If the agents interact, their interaction partners may be selected randomly, and so on. The same is presumably true of the social world: There is a degree of random chance in what happens. The effect of this is that running the model a number of times will yield different results each time (this is dealt with in more detail in the next section). Even if the results are only slightly different, the best one can hope for is that the most frequent outcome—the mode of the *distribution* of outputs from the model—corresponds to what is

actually observed (Axelrod, 1997a; Moss, 2002). If it does not, one might wonder whether this is because the particular combination of random events that occurred in the real world is an outlier and, if it were possible to "rerun" the real world several times, the most common outcome would more closely resemble the outcome seen in the model!

3.4 Techniques for Validation

Two areas need to be examined when validating models: first, the fit between a theory and the model of that theory, and second, the fit between the model and the real-world phenomenon that the model is supposed to simulate.

3.4.1 Comparing Theory and the Model: Sensitivity Analysis

The fit between a theory and its model is best evaluated by using the theory to derive a number of propositions about the form of the relationships expected between variables and then checking whether the expected distributions do, in fact, appear when the model is run using a variety of parameter settings. Each of the parameter settings corresponds to an assumption made by the model. One should aim to check each of the settings either by measuring the value from empirical data or by conducting a *sensitivity analysis*. Although the former is preferable, there will be many parameters that cannot be checked empirically, and for these, some form of sensitivity analysis is essential.

Sensitivity analysis is aimed at understanding the conditions under which the model yields the expected results. For example, with the opinion dynamics model described in Chapter 1, one might ask, how extreme do the extremists have to be for all the agents eventually to join one of the extreme parties? To find out, one needs to run the simulation for a series of values of the uncertainty parameter, perhaps ranging from 0.5 to 1.0 in steps of 0.1 (i.e., six runs). But the model includes random elements (for example, the order in which agents "meet" and exchange opinions is random), so one should not be content with just six runs, but should perform a number of runs for each value of the uncertainty parameter to obtain a mean and variance. If one does 10 repetitions for each parameter setting (the number needs to be chosen with the amount of variation in mind, so that one gets a statistically meaningful result), we would need to carry out 60 runs.

To make matters worse, most models include many parameters, and their interaction may affect the simulation (e.g., the number of extremist groups that emerge depends on both the uncertainty of the extremists and the distribution of extremists across the political spectrum, the parameters having an

effect both independently and in combination), so that ideally one would want to examine the output for all values of all parameters in all combinations. Even with only a few parameters, this can require an astronomical number of runs and thus is not a practical strategy.

If we think of the range of each parameter as lying on an axis, the set of all parameters defines a multidimensional parameter space, in which each point corresponds to one combination of parameter values. The scale of the task in doing a full sensitivity analysis can then be quantified as the volume of this space, and any way of cutting down the space will reduce the number of simulation runs needed. One obvious way is to use prior empirical knowledge to restrict the range of as many parameters as possible. For instance, we might know that a parameter, although theoretically capable of taking any value between 0 and 100, in fact is never observed to have a value greater than 10. Alternatively, we can limit the applicability of the model by constraining the range of values we test: We may state that the model applies only if the parameter is somewhere between 5 and 10 and not investigate what happens when it is outside this range.

Another approach, which can be used in combination with limiting the range of parameters, is to sample the parameter space. Instead of performing simulation runs at every point in the space, we use only some points. These points may be chosen randomly or purposively to inspect combinations that we think are particularly interesting or that are close to regions where major changes in the simulation's behavior are expected (*phase changes*).

A sophisticated version of this approach uses a learning algorithm, such as the genetic algorithm (see Section 5.2.2) to search the space to identify regions where some output variable or variables take their maximum or minimum values (Chattoe, Saam, & Möhring, 2000).

3.4.2 Comparing the Model and Empirical Data

As discussed in the previous section, not all models are expected to match empirical data; there may be no data available to compare with models whose objective is the development of theory, and no reason to conduct empirical tests. For middle range models, the criterion is whether the simulation generates outputs that are qualitatively similar to those observed in the social world, but a quantitative match is not expected. It is only with what we have called facsimile models that there are stringent requirements for comparison between data obtained from the simulation and empirical data. This section considers some of the ways of doing this.

Social scientists are well used to comparing data obtained from models and data collected from the social world: This is what is being done implicitly every time one calculates an R^2 (the coefficient of determination) in

an ordinary linear regression (Fielding & Gilbert, 2000). The model in this case is the regression equation, which computes the predicted values of the dependent variable. Similarly, one can measure the fit between the values of a variable that are output from a simulation model and the values observed empirically (this is, in fact, just the correlation coefficient between the two sets of values). However, this simple procedure makes a number of strong assumptions that, although often satisfied for linear regressions, are much less likely to be appropriate for simulation models.

One important characteristic of simulation models is that the values of output variables change as the simulation runs. For example, in a model of consumer behavior, the number of purchasers of a particular brand might be observed growing from zero to a majority during a simulation. The growth trend might then be compared with the growth in sales of an actual product. Because the data to be compared are time series, one must allow for the fact that there is autocorrelation: The value at time $t + 1$ is not independent of the value at time t. Statistical procedures called ARIMA can be used to compare such time series (Chatfield, 2004).

3.5 Summary

In this chapter, a number of conceptual issues involved in designing and carrying out agent-based modeling research have been considered. We have shown by example the type of analysis that needs to be done before one begins programming and have mentioned some of the challenges that are raised by wanting one's model to be verifiable and valid. The next chapter will move on to matters of implementation: how one can code a model; plan its development; and, finally, report its results.

4. DESIGNING AND DEVELOPING AGENT-BASED MODELS

4.1 Modeling Toolkits, Libraries, Languages, Frameworks, and Environments

Although some modelers build their agent-based models using only a conventional programming language (most frequently Java, although any