Chapter 8
Validating Simulations

Nuno David

Why Read This Chapter? To help you decide how to check your simulation – both against its antecedent conceptual models (verification) and external standards such as data (validation) – and in this way help you to establish the credibility of your simulation. In order to do this the chapter will point out the nature of these processes, including the variety of ways in which people seek to achieve them.

Abstract Verification and validation are two important aspects of model building. Verification and validation compare models with observations and descriptions of the problem modelled, which may include other models that have been verified and validated to some level. However, the use of simulation for modelling social complexity is very diverse. Often, verification and validation do not refer to an explicit stage in the simulation development process, but to the modelling process itself, according to good practices and in a way that grants credibility to using the simulation for a specific purpose. One cannot consider verification and validation without considering the purpose of the simulation. This chapter deals with a comprehensive outline of methodological perspectives and practical uses of verification and validation. The problem of verification and validation is tackled in three main topics: (1) the meaning of the terms verification and validation in the context of simulating social complexity; (2) methods and techniques related to verification, including static and dynamic methods, good programming practices, defensive programming, and replication for model alignment; and (3) types and techniques of validation as well as their relationship to different modelling strategies.

N. David (✉)
ISCTE – Lisbon University Institute, Av. das Forças Armadas, Lisbon 1649-026, Portugal
e-mail: Nuno.David@iscte.pt

B. Edmonds and R. Meyer (eds.), *Simulating Social Complexity*,
Understanding Complex Systems, DOI 10.1007/978-3-540-93813-2_8,

8.1 Introduction

The terms verification and validation (V&V) are commonly used in science but their meaning is often controversial, both in the natural and the social sciences. The purpose of this chapter is not to describe any general theory of model V&V. A general theory of that kind does not exist.

Besides the epistemological underpinnings of the terms, their use in simulation has a pragmatic nature. In disciplines that make use of computerised models, the role of V&V is related to the need of evaluating models along the simulation development process. Basically, the very idea of V&V is comparing models with observations and descriptions of the problem modelled, and this may include other models that have been verified and validated to some level. This chapter describes methodological perspectives and practical uses of the terms as well as different strategies and techniques to verify and validate models of social complexity, mostly in social simulation.

The use of simulation for modelling social complexity is very diverse. Often, V&V do not refer to an explicit stage in the simulation development process, but to the modelling process itself according to good practices and in a way that grants credibility to using the simulation for a specific purpose. Normally, the purpose is dependent on different strategies or dimensions, along which simulations can be characterized, related to different kinds of claims intended by the modeller, such as theoretical claims, empirical claims or subjunctive theoretical claims. The term subjunctive is used when simulations are used for talking about scenarios in possible worlds, such as describing 'what *would* happen if something were the case'. There cannot be V&V without considering the purpose of the simulation.

In the next section of the chapter, I will deal with the meaning of the terms V&V in the context of the simulation development process. While the terms are often used with the same meanings, or even interchangeably, practical reasons exist for distinguishing between them. Whereas verification concerns the evaluation of the implementation of the model in terms of the researchers' intentions, validation refers to the evaluation of the credibility of the model as a representation of the subject modelled. In Sect. 8.3, methods and techniques related to verification are described. Subsequently, in Sect. 8.4, validation types and techniques are described, as well as their relationship to different modelling strategies.

8.2 The Simulation Development Process

Several chains of intermediate models are developed before obtaining a satisfactory verified and validated model. What does it mean to verify and validate a model in social simulation? Is there a fundamental difference between verifying and validating models? The purpose of this section is to define the role of V&V within the scope of the simulation development process.

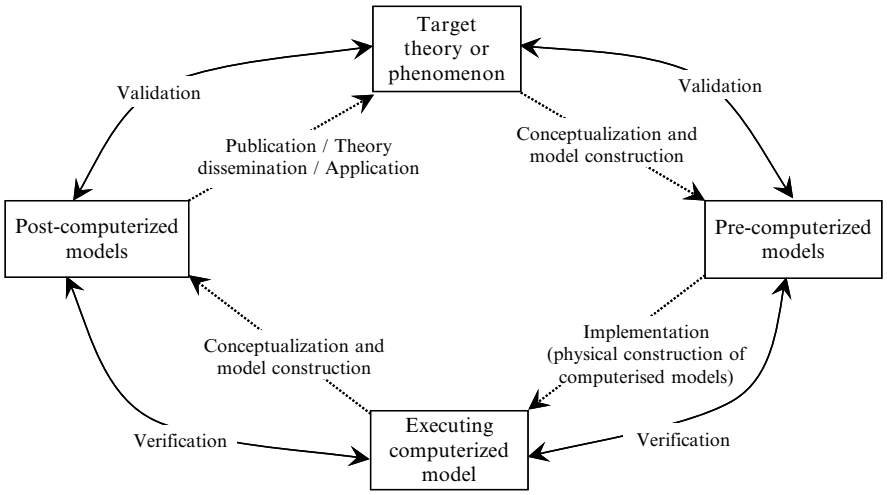

Fig. 8.1 Verification and validation related to the model development process (David 2009)

The most common definitions of V&V are imported from computer science, as well as from technical and numerical simulation,[1] having intended distinct – although often overlapping – meanings. The reason for distinguishing the terms is the need to determine the suitability of certain models for representing two distinct subjects of inquiry. This is represented in Fig. 8.1, in which V&V are related to a simplified model development process. Two conceptual models mediate between two subjects of inquiry. The latter are (1) the target theory or phenomenon and (2) the executing computerised model. The conceptual model on the right, which is designated here as the *pre-computerised model*, is basically a representation in the minds and writing of the researchers, which presumably represents the target. This model must be implemented as a *computerised executable model*, by going through a number of intermediate models such as formal specification or textual programs written in high-level programming languages.

The analysis of the executing model gives rise to one or more conceptual models on the left, which are designated here as the *post-computerised* models. They are constructed based on the output of the computerised model, often with the aid of statistical packages, graphing and visualisation. The whole construction process results in categories of description that may not have been used for describing the pre-computerised model. This is the so-called idea of *emergence*, when interactions among objects specified through pre-computerised models at some level of description give rise to different categories of objects at macro levels of description observed in the executing model, which are later specified through post-computerised models.

[1] Numerical simulation refers to simulation for finding solutions to mathematical models, normally for cases in which mathematics does not provide analytical solutions. Technical simulation stands for simulation with numerical models in computational sciences and engineering.

As an example consider the culture dissemination model of Axelrod (1997a) which has a goal of analysing the phenomena of social influence. At a micro-level of description, a pre-computerised model defines the concept of *actors* distributed on a grid, the concept of *culture* of each actor as a set of five features and the *interaction mechanisms* specified with a bit-flipping schema, in which the probability of interaction between two actors is set proportionately to measure the similarity between two cultures.

The executing model is then explored and other categories of objects resulting from the interaction of individual cultures may be defined, associated with macro properties of interest and conditions in which they form, such as the concepts of *regions* and *zones* on the grid. A great deal of the simulation proposed by Axelrod concerns investigating properties of *regions* and *zones* in the context of the conceptual, post-computerised, model proposed, such as the relation between the size of a *region* formed and the number of features per *individual culture*. These concepts are interpreted in relation to the target social phenomena of social influence.

I will now situate the role of V&V in the modelling process of social simulation.

8.2.1 What Does It Mean to Verify a Computerised Model?

Computerised model *verification* is defined as checking the adequacy among conceptual models and computerised models (see also Chap. 6 in this volume, Galán et al. 2013). Consider the lower quadrants of Fig. 8.1. They are concerned with ensuring that the pre-computerised model has been implemented *adequately* as an executable computerised model, according to the researchers' intentions in the parameter range considered, and also that the post-computerised model *adequately* represents the executing model in the parameter range considered.[2]

At this point you might question the meaning of *adequately*. There is no formal definition of adequately. Actually, it is a rather difficult epistemological problem, considered in the series of EPOS meetings about epistemological perspectives on simulation (see Frank and Troitzsch 2005; Squazzoni 2009; David et al. 2010). A practical and minimal definition could be the following: adequateness means that the relationship between inputs and outputs of the computerised model is consistent with the semantics of both the pre- and post-computerised models, in accordance with the researcher's intentions. However, the outcomes of computer programs in social simulation are often unintended or not known a priori and thus the verification process requires more than checking that the executing model does what it was planned to do. The goal of the whole exercise is to assess logical links within, as well as between, the pre- and the post-computerised models. This requires assessing whether the post-computerised model – while expressing concepts that the pre-computerised model does not express – is consistent with the latter. From a methodological point of view

[2] Verification in the left quadrant of Fig. 8.1 is sometimes known as "internal validation".

this is a complicated question, but from a practical perspective one might operationally define the verification problem with the following procedures:

(a) For some pre-computerised model definable as a set of input/output pairs *in a specified parameter range*, the corresponding executing model is *verified for the range considered* if the corresponding post-computerised model expresses the same set of inputs/outputs for the range considered.
(b) For some pre-computerised model defined according to the researcher and/or stakeholders' intentions *in a specified parameter range*, the corresponding executing model is *verified for the range considered* if the corresponding post-computerised model complies with the researchers and/or stakeholders' intentions for the range considered.

Note that both procedures limit the verification problem to a clearly defined parameter range. The first option is possible when extensive quantitative data is available from the target with which to test the executing model. This is normally not the case and the verification problem often amounts to the last option. This is possible since the aim of verification is to assess the appropriateness of the logical links that may be established between micro-levels of description specified in the pre-computerised model and macro-levels of description specified through post-computerised models, which should be amenable to evaluation by researchers and stakeholders. Nevertheless, as we will discuss further, a computerised model should only be qualified as verified with reasonable confidence if it has been successfully subjected to a procedure known in software engineering as *N-version programming*. A synonym commonly used in social simulation consists of *replicating implementations* for model alignment. This is described in Sect. 8.2.4.

It is important to observe that, unlike many areas of computer science and formal logics, the verification process should not be understood as a mere process of evaluating the correctness of formal inference steps. The coherence among conceptual and computerised models, in the light of the researchers' and stakeholders' intentions, is, to a large extent, narratively evaluated. Were we to envisage computers and programming languages as mere formal machines then computers and programming languages would have limited expressiveness for representing inferences upon semantically rich descriptions of social processes. After defining the meaning of validation in the following section, this will be a point where the meanings of verifying and validating a simulation model will overlap, and it may not be trivial to distinguish between the two.[3]

[3] From a technical point of view, in classical computer theory, verification amounts to ascertaining the validity of certain output as a function of given input, regardless of any interpretation given in terms of any theory or any phenomenon not strictly computational – it is pure inference in a closed world. But this would require us to assume that social processes are computational in a Church-Turing sense, which seems difficult to conceive. For an elaboration on this see (David et al. 2007).

8.2.2 What Does It Mean to Validate a Model?

Model *validation* is defined as ensuring that both conceptual and computerised models are adequate representations of the target. The term "adequate" in this sense may stand for a number of epistemological perspectives. From a practical point of view we could assess whether the outputs of the simulation are close enough to empirical data.

We could also assess various aspects of the simulation, such as if the mechanisms specified in the simulation are well accepted by stakeholders involved in a participative-based approach. In Sect. 8.4 we will describe the general idea of validation as the process that assesses whether the pre-computerised models – put forward as models of social complexity– can be demonstrated to represent aspects of social behaviour and interaction able to give rise to post-computerised models that are, at some given level, consistent with the subjacent theories or similar to real data.

Given the model development process described, is there any practical difference between verifying and validating simulations? Rather than being a sharp difference in kind it is a distinction that results from the computational method. Whereas *verification* concerns the assessment of the logical inferences that are established between micro and macro concepts with close reference to the computerised model, *validation* concerns the evaluation of such inferences and concepts with a closer reference to the target.

In paraphrasing Axelrod (1997b), at first sight, we could say that the problem is whether an unexpected result is a reflection of the computerised model, due to a mistake in the implementation of the pre-computerised model, or is a surprising consequence of the pre-computerised model itself. Unfortunately, the problem is more complicated than that. In many cases mistakes in the code may not be qualified simply as mistakes, but only as one interpretation among many other possible interpretations for implementing a conceptual model. Nevertheless, from a practical viewpoint there are still good reasons to make the distinction between V&V. A number of established practices exist for the corresponding quadrants of Fig. 8.1. We will address these in the following sections.

8.3 Verification Methods and Techniques

The need to transform conceptual models into computerised models and back to conceptual models is one of the substratums of simulation. This process is illustrated in Fig. 8.2. Several conceptual models are constructed and specified in order to obtain a proper implementation as a computerised model. The intended computerised model is specified as a computer program in a readable form with a high-level language. The compilation of the program into a low-level program, followed by its execution with a set of inputs, results in outputs described with data and visualization models. Given the use of pseudo-random generators, the simulation will be run many times with the

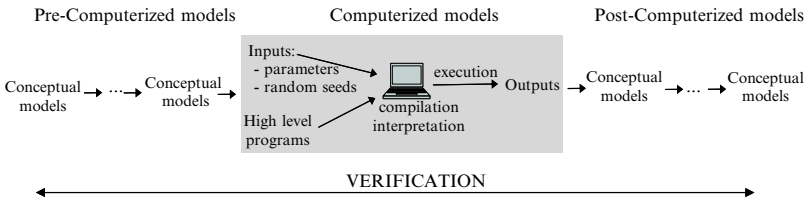

Fig. 8.2 Implementation of pre-computerised models and construction of post-computerised models

same inputs. At any rate, the complexity of social theories and phenomena implies that the simulation must be run many times in order to observe its behaviour in a variety of conditions. A post-computerised model is built, i.e. a conceptual model describing the simulation behavior, possibly incorporating new concepts and knowledge not captured with the pre-computerised model.

8.3.1 Static and Dynamic Methods

In simulation, verification techniques involve showing that the process of translation among conceptual and computerised models is carried out according to the researchers' intentions. Computer science, mostly in the field of software engineering, has developed numerous approaches and techniques for writing programs and verifying their execution in computers. Most of these are not particular to simulation and not all are well suited to social simulation. The techniques and terminology that we describe in this section are based on Sommerville (1995) and Sargent (1999), and are adapted to social simulation whenever appropriate. There are two basic approaches for verification: *static and dynamic methods*.

Dynamic methods involve exercising the computerised model with sets of inputs and the results obtained are used to determine whether the computerised model has been programmed appropriately. A possible *simulation failure* occurs when the simulation is executing and does not behave as expected. Whether the unexpected output is indeed a failure or a legitimate consequence of the pre-computerised model concerns the use of *static methods*. Static methods involve showing that the computerised model appropriately implements the pre-computerised models without *software faults*, i.e. without programming errors whereby the computerised model does not conform to the researchers' intentions.

Software faults are static and their existence may be inferred from simulation failures and further inspections to the high-level program code. A software fault causes a simulation failure when the faulty code is executed with a set of inputs that expose the simulation fault (cf. Sommerville 1995).

We can never conclusively demonstrate that a simulation is fault free, but we can increase our confidence in a program *within the range of the parameter space tested* by adopting good programming and testing practices in order to minimize faults.

Good practices common to any kind of programming include commenting your code and testing the program with parameter values whose outputs are known or with values that are at the extremes of the legal values.

Nevertheless, not all programming and testing techniques in computer science can be applied to social simulation. For the most part, simulation is used to explore concepts that are not anticipated during the specification of pre-computerised models. Virtually, the software notion of "functional requirement" does not exist in social simulation. If we are dealing with complex social models, it is virtually impossible to enumerate a priori an exhaustive list of requirements that a program should satisfy (David et al. 2003). To the extent that the researcher does not know what to expect, program testing is not enough to show that the program properly reflects the intended model.

In any case, the verifiability of a simulation is influenced by the process used to develop that simulation. The adoption of good programming practices for designing and implementing the model is fundamental. Defensive programming methodologies, like contract-based programming with use of assertions, are well suited for the explorative nature of simulation. Defensive programming consists of including extra, redundant, code to test for errors and to aid in general debugging. In addition, the need to produce several implementations of the same conceptual model is increasingly playing a role. These techniques are described below along four major topics: good programming practices, defensive programming, replication, and a very brief reference to participative-based methods.

8.3.2 Good Programming Practices

Good programming in social simulation includes several approaches, techniques or mechanisms to improve verifiability in general and, specifically, to decrease the number of expected simulation faults, to ease debugging and to improve code readability as well as faster and more flexible development. Available techniques and mechanisms include modularity, encapsulation, high-level memory management and, more generally, software reuse. Most object-oriented programming languages include built-in mechanisms that simplify their use. The following deserve particular attention.

8.3.2.1 Modularity and Encapsulation

Programs should be written and verified in modules or subprograms.[4] In the words of Kleijen (1995), this is a kind of "divide and conquer" process, where one verifies the whole simulation module by module. Modular programming simplifies the

[4] We consider "modules" and "sub-programs" as synonymous.

location of faults in the program. A module is any component that provides one or more services to other modules and which separates its public interface from its implementation details. The public interface of a module concerns the specification for how other modules may request services from it. The implementation concerns particular design decisions hidden within the module and not accessible to the other modules, which are more liable to faults or design changes along the simulation development process. If the implementation of a module is changed this should not affect how other modules request services from it. A typical kind of module is a *class* in object-oriented programming.

A class is a model for object instantiation. Objects are independent, loosely coupled entities in which the particular implementation of the object state (information representation) and of the services provided by the object (information processing) should not affect the public interface through which other objects request services from it. An object comprises a set of private attributes – which define its state – and a set of public operations that check or act on those attributes – which define its interface. Insofar as the public interface remains the same along the development process, the internal data representation of the class may be changed along the way without affecting the way other classes request services from it. The grouping of related classes within a single file is another kind of module, usually called a physical module. Yet another kind is the concept of *package*, defined as a group of related files, such as in the Java programming language. The public interface of a package is the set of all public services available in all files. The package design details most likely to change are hidden behind the interface, possibly implemented with the aid of other private classes in the package.

It is essentially up to the programmer to decide which data structures, classes and files should be grouped together. According to Sommerville (1995, pp. 218–219), the *cohesion* of a module is a measure of the closeness of the relationship between its components. Conversely, the *coupling* among modules measures the strength of interconnections among them. As a rule, modules are tightly coupled if they make use of shared variables or if they interchange many types of control information. A good design calls for *high cohesion within* a module and *low coupling among* modules. For instance, an agent implemented as a group of classes should have its public interface clearly defined and be as independent as possible from the group of classes that implement the interaction environment of the agents, such as a bi-dimensional torus grid. Whether the grid is implemented as a torus or as a network should not imply future changes to the implementation of the agent architecture, which in any case would request the same services from the grid, such as moving to the left or right. Modularity encourages the production of readable and testable simulations. Moreover, insofar as agent-based modelling can be easily mapped to the programming of interacting modules, the adoption of bottom-up design and testing approaches is a natural way to proceed. Small modules at the lower levels in the hierarchy are tested first and the other modules are worked up the hierarchy until the whole simulation is tested.

8.3.2.2 High-Level Memory Management

The use of pointers and pointer arithmetic in a program are low-level constructs available in some high-level languages that manipulate variables referring directly to the machine memory. They are inappropriate in social simulation programs because they allow memory leaks and foment aliasing, which makes programs prone to bugs. A memory leak occurs when a program fails to release memory no longer needed, which may have the effect of consuming more memory than necessary and decreasing efficiency. We say there is aliasing when an object is changed through a reference variable with unwanted indirect effects on another variable. Both leaks and aliasing make programs harder to understand and faults harder to find. Whenever possible, high-level languages with pointer arithmetic should be avoided. In programming with object-oriented languages, the use of built-in mechanisms for automatic memory management, responsible for managing the low-level lifecycle of objects in memory, should be preferred. Whereas it is up to the programmer to determine the points where objects are created in the program, mechanisms of memory management free the programmer from the trouble of deleting the objects from memory at the points where they become useless. Fortunately, most programming languages used in current agent-based simulation toolkits are examples of high-level languages that have built-in mechanisms for automatic memory management.

8.3.2.3 Software Reuse

We sometimes tend to prefer rewriting modules insofar as we believe that our modules are better programmed than others. However, software modules that are used and tested in a variety of different situations have fewer faults than modules developed for the purposes of a single simulation. A good programmer should not assume that all modules should be implemented especially for the simulation being developed. Reusable modules that have been subjected to previous use and verification should be preferred to those built from scratch. The more a simulation is based on reusable modules the fewer modules need to be specified, implemented and verified.

Software reuse is used in social simulation as more and more developing frameworks become available. Module reuse in simulation is basically a type of model embedding. The use of agent-based toolkits as special-purpose extensions to standard high-level languages based on Java, Objective C, or SmallTalk provides developing standards and faster development, making simulations more comparable to each other. Software reuse in computer science and engineering can be considered at a number of different levels (Sommerville 1995, p. 397). Likewise in social simulation we may identify different levels from the strongest to the weakest degrees of dependency on the specifics of simulation platforms and toolkits. In this sense, strong reuse means constraints on the simulation to existing models and weak reuse lets the researcher develop models more freely.

Model Architecture Level

The control subsystem of a simulation, such as a discrete-time scheduling mechanism, may be reused. In most agent-based simulation toolkits, scheduling consists of setting up method calls on objects to occur at certain times. Scheduling mechanisms are thus part of a reusable agent skeleton template, extended according to the specifics of each agent model. Other examples include whole collections of reusable objects and mechanisms, such as interaction environments that act as agent containers and define the interactive relationship of agents relative to each other, such as a network model or a torus grid. Yet other examples may include entire canonical simulation models, which are used as components of another extended, more sophisticated, simulation model.

Module or Object Level

Components representing a set of functions may be reused, such as class libraries for generating sequence graphs; histograms or plots for visualization models; statistical packages for data analysis; and widely tested pseudo-random number generators that aggregate a set of different random number distributions into a single class or package.

Programming Language API Level

Standard libraries and Application Programming Interfaces (APIs) conventionally available in every implementation of high-level languages may and should be used. Typically, abstract data structures representing such things as lists, queues and trees, are coupled in a set of classes that implement reusable data structures, such as the Java Collections Framework.

Generic-Type Level

The use of generic classes is a kind of reuse. Generics are a way of creating parameterized general purpose templates for class types and subroutines (see e.g. Sommerville 1995). If the same kind of abstract structure, such as a list, is used for many different types of elements, like a list of agents and a list of events, this avoids the need to develop a separate implementation of the list for each type of element. As a result, programmers are less likely to make mistakes, thus avoiding faults and improving confidence in the simulation. Generic programming facilities are available in several object-oriented languages, such as C++ and Java.

Routine Level

Independent components implementing a single procedure or function may be reused, for example, a mathematical function used in a previously tested and verified setting can be reused in another simulation.

8.3.3 Defensive Programming

The focus of good programming practices is on good code readability, flexible and fast development as well as ease of debugging. Other focuses yet to be considered are the actual testing and location of faults in the program. A typical test consists of running test cases for which outputs are known and confirming these every time the program is changed. Defensive approaches require that the programmer should never assume that a module will work as expected but instead the programmer should handle the appropriate testing for every module, possibly embedding the test in the program itself. Rather than considering testing only after programming the model, some of the testing should be planned and embedded as redundant code in the program. In software engineering defensive programming stands for an approach to fault tolerance for failure recovery. In social simulation the goal is somewhat different and consists of including extra, redundant, code to test for program failures and assist in the location of program faults.

Two techniques seem relevant to social simulation: contract-based programming and unit testing. Although the latter is often referred to in the literature, it is not a common practice in social simulation, probably because it requires a considerable amount of redundant code to test programs.

The former stands for a methodology for checking the violation of specified conditions in order to verify programs written with procedural languages on the basis of declarative statements, called *assertions*, which are appropriate to the kind of explorative programming used in simulation.

Testing and contract-based programming can be combined with the use of assertions and *exceptions*. Assertions are predicates placed in the program to test whether a specified condition is satisfied during testing. If the predicate evaluates to false, the program should halt for further debugging. Exceptions are messages that indicate the occurrence of exceptional conditions that were anticipated by the programmer, which usually halt the program as well.

Contract-based programming is a methodology for designing class specifications with checkable conditions at run time. The attributes of a class define the state space for the objects of that class. The set of states considered valid act as constraints on the range of values that objects of that class can assume. Constraints are limits defined over the ranges of the attributes of a class and may define dependencies among those attributes. If the valid state space of a class is violated in execution time then an exceptional condition is launched and the program execution halts, indicating information about the location and the type of violation occurred.

Constraints can also be specified at the routine level by specifying the range of the parameter values and the dependencies among parameters. In fact, the specification of a class may be understood as defining a *contract* between two programmers, the *client* and the *supplier* of that class, resulting in two types of constraints:

1. The methods' pre-conditions specified by the supplier, to which the client should abide;
2. The class invariant and the methods' post-conditions to which the supplier commits.

If the pre-conditions are not satisfied, then an exceptional condition occurs, which must be dealt with by the client programmer who violated the contract of that class. In contrast, if the class invariant or the post-conditions are not satisfied, then the fault is due to the particular implementation of that class, resulting in an assertion failure. This means that the supplier may not have programmed the class appropriately.

As an example, consider a partial definition of a class named Agent that specifies an agent in a culture dissemination model. Agents are distributed on a grid and the culture of each actor is defined as a nonempty set of features with a positive number of traits per feature. Each agent is aware of its position on the grid, which cannot be outside the grid bounds. The attributes of that class include a list of features, the number of traits per feature and the agent's position. The invariant of such a class would be defined as follows:

$$\text{features} \neq \{\} \textit{ and } \text{number\_traits} > 0 \textit{ and } \text{position is legal}$$

Suppose that agent interaction is specified with a bit-flipping kind of schema, which occurs only between contiguous neighbors and if the agents have different traits in at least one feature. Moreover, the agents must share the same trait in at least one feature after the interaction. In order to guarantee that every state change in objects of type Agent is valid, the invariant must be checked (1) right after the calling of that method; (2) right before it returns to the caller; and (3) right before the end of the class constructors. That is, whereas the bit-flipping pre-conditions should be checked *before* the interaction takes place, the post-conditions should be checked *after* the interaction. A partial definition of a Java class could be the following (with redundant code in boldface):

```
public class Agent {

    private SpaceGrid grid; // the agent´s interaction
    environment
    private int [] features;
    private int number_of_traits;
    private Position position; // the agent's current
    position in the grid
```

```
// class constructor
public Agent(final SpaceGrid grid, final int [] features,
             final int number_of_traits, final Position
             position) {

    this.grid = grid;
    this.features = features;
    this.number_of_traits = number_of_traits;
    this.position = position;

    // check the class invariant after initialisation
    assert checkInvariant();
}

// other methods here

// the bit-flipping operation
public void bitFlip(final Agent theOtherAgent) {

    // check the invariant before every method that
    change the agent's state
    assert checkInvariant();

    // test pre-conditions for bit-flipping
    if (theOtherAgent == null)
        throw new NullPointerException();
    if (!grid.areNeigbourhs(this, theOtherAgent)) //
    are the agents neighbours?
        throw
        new IllegalArgumentException("Agents are not
        neighbours, cannot bit-flip");
    // agents must have a different trait in at least one
    feature
    if (Arrays.dontMatch(features, theOtherAgent.
    getFeatures()) < 1)
        throw
        new IllegalArgumentException("Insufficient
        number of features to bit-flip");

    // make bit-flipping

    int[] other_features = theOtherAgent.getFeatures();

    int rand = randomFeature(features);
    while (features[rand] == other_features[rand])
       rand = randomFeature(features);
    other_features[rand] = features[rand]; // bit-flip
```

```
        // test post-conditions (agents must share at least
        one feature after bit-flip)
        assert Arrays.match(features, theOtherAgent.
        getFeatures()) >= 1;

        // check the invariant after every method that
        changed the agent's state
        assert checkInvariant() ;
    }

    private boolean checkInvariant() {

        if (grid == null || features == null || position ==
        null ||
            number_of_traits < 1 || grid.outOfBounds
            (position))
            return false;

        return true;
    }
} // end of class Agent
```

In the program, the invariant is checked at the end of the constructor, right after initializing the agent's attributes, and both before and after any state change in the methods. Since the invariant concerns the testing on the part of the programmer of that class (the supplier), rather than the user of that class (the client), the method checkInvariant is set to private. Since the method bitFlip changes the agent's state, the method pre-conditions and post-conditions should be checked. If the pre-conditions are not satisfied, an exception is thrown. This means that the error is on the caller of that method. If the class invariant or the post-conditions are not satisfied, then an assertion failure is launched, suggesting an error in the implementation of that class.

Contract-based programming helps programmers reason about the conceptual model and its programs. Too often it is difficult to understand the details of an implementation. Insofar as the contracts of a module are fully documented and are regarded as a precise specification of the computerised model – akin to a declarative programming language style – additional benefits are increased readability and the fostering of replicability. During testing the programmer will typically run the program with assertions enabled. For reasons of efficiency, the assertion mechanism can either be set to on or off.

8.3.4 Replication for Model Alignment

Program testing is an essential activity for building simulations. However, testing alone cannot distinguish between faults in the implementation of the pre-computerised model and a surprising consequence of the model itself. Static methods are thus unavoidable and the examination of the source code is important in all stages of the

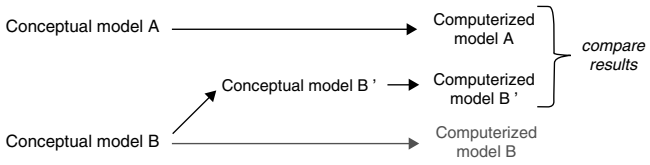

Fig. 8.3 Model alignment

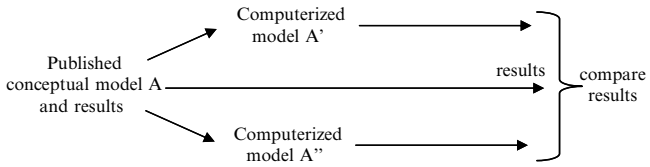

Fig. 8.4 Model replication

development process. In practice, static and dynamic methods are used together. Most integrated development environments provide debugging tools and allow the programmer to trace the execution of a program and observe the values of variables statement by statement.

A slightly different technique, mostly used in numerical and technical simulation, is called "structured walk-throughs." This consists of having more than one person reading and debugging a program. All members of the development team are given a copy of a particular module to be debugged and the module developer goes through the code but does not proceed from one statement to another until everyone is convinced that a statement is correct (Law and Kelton 1991).

A different technique described by Sargent (1999) consists of reprogramming critical modules to determine if the same results are obtained. If two members are given the same specification for a module and they make two distinct implementations and obtain the same results, then the confidence in the implementation is increased. A more general problem is the extent to which models can be related to others so that their consequences and results are consistent with each other. In its most general form, this concerns both to V&V. After Axtell et al. (1996) it became known as the process of *model alignment*, which is used for determining whether different published models describing the same class of social phenomena produce the same results. Usually the alignment of two models A and B requires modifying certain features of model B – for instance by turning off certain features – in order to become equivalent to model A. This is represented in Fig. 8.3.

The term "model alignment" is frequently used synonymously for *model replication*. This assesses the extent to which building computerised models that draw on the same conceptual, usually published, model give results compatible with the ones reported in the published model. If the new results are similar to the published results, then the confidence in the correspondence between the computerised and the conceptual models is increased. Replication is represented in Fig. 8.4.

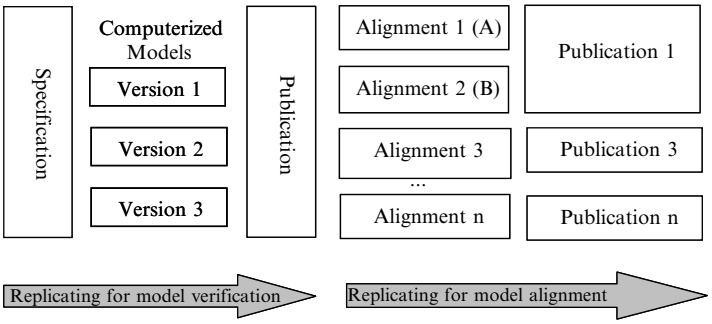

Fig. 8.5 N-version programming and replication for model alignment

In the vast majority of cases, replication processes are mostly reactive and not proactive, in other words, the creator of the model publishes it and in the future someone else replicates it. However, we should not take replication lightly and see it as something that will possibly be done in the future. A verification technique, not so frequently used in social simulation, is called *N-version programming*. This is similar to reprogramming critical modules. In contrast to the practice of having other teams replicating models after they have already been reviewed, accepted and published, the effort of replication becomes focused on producing multiple computerised versions of a conceptual model *before* submitting it for peer reviewing.

Both N-version programming and replication are depicted in Fig. 8.5. The right-hand side of the diagram illustrates a perspective where a published conceptual model is replicated. The left-hand side illustrates a perspective where a conceptual model gives origin to multiple computerised versions of the model, implemented by different persons before it is actually published. In any case, if all versions lead to the same results, then there are reasonable grounds for trusting in the results obtained, as well as in the correspondence between the conceptual and computerised models.

8.3.4.1 Types of Model Equivalence

The work of Axtell et al. (1996) is arguably the most-cited attempt to align two distinct but similar models. Rather than re-implementing Axelrod's culture dissemination model, Axtell and colleagues focused on the general case of aligning two models that reflected slightly distinctive mechanisms. For this purpose, Epstein and Axtell's Sugarscape model was progressively simplified in order to align with the results obtained by Axelrod's culture dissemination model.

Model alignment has been further investigated in a series of meetings called model-to-model (M2M) workshops (see Rouchier et al. 2008). The M2M workshops attract researchers interested in understanding and promoting the transferability of knowledge between model users. The replication of Edmonds and Hales (2003) is particularly informative on the problem of verification. They suggested that one should not trust an unreplicated simulation, since its results are almost certainly wrong in the sense that the computerised model differs from what was intended or

assumed in the conceptual model. In order to align with a published conceptual model they had to rely on a double replication process insofar as the first replication did not seem to result in alignment. It was only after implementing a second computerised version of the published model that the authors were sure that the results reported in the published model did not align with their own. They concluded that the original implementation of the published model was producing results that could be misleading. This was due to different interpretations of the correct implementation of a mechanism of tournament selection for reproducing agents. Subtle differences in the mechanism implied different conclusions about the functioning of the model.

But how do we determine whether or not two models produce equivalent results? Axtell et al. (1996) defined three kinds of equivalence:

- Numerical identity: shows that the two models reproduce the results exactly.
- Relational equivalence: shows that the two models produce the same internal relationship among their results, for instance that a particular variable is a quadratic function of another.
- Distributional equivalence: shows that the two models produce distributions of results that cannot be distinguished statistically.

Numerical identity is hardly attainable in social complexity, except in the circumstances where models converge to some kind of final equilibrium, such as in Axelrod's culture dissemination model. Most simulations display several kinds of global behaviours that do not converge to equilibrium, are in the medium term self-reinforcing and make numerical identity unattainable. Among the three kinds of equivalence, distributional equivalence is the most demanding: it is achieved when the distributions of results cannot be distinguished statistically. What this shows is that at conventional confidence probabilities the statistics from different implementations *may* come from the same distribution, but it does *not prove* that this is actually the case. In other words, it does not prove that two implementations are algorithmically equivalent, but it allows us to disconfirm that they are. If this test survives repeatedly we somehow increase our confidence in the equivalence but only in the parameter range within which it has been tested.

8.3.4.2 Programming for Replication

An important aspect when programming a simulation is to guarantee that it may be replicated by other researchers. Designing and programming for replicability involves a number of aspects that should be considered. Simulations are often a mix of conceptual descriptions and hard technical choices about implementation. The author who reports a model should assume that an alignment may later be tried and thus should be careful about providing detailed information for future use:

- Provide effective documentation about the conceptual and the computerised models; provide information about those technical options where the translation from the conceptual to the computerised model is neither straightforward nor consensual. Even if the difference between two computerised models may seem

minor or irrelevant, small differences can affect the level of equivalence, even if the overall character of the simulation does not seem to change significantly.

- Outline in pseudo-code the parts of the program that may be liable to ambiguities; use modelling languages to represent implementation options, like UML. This is a formalism used to describe not only the static structure of the relations between classes but also different aspects of its dynamic behaviour, for instance, with activity graphs and sequence diagrams.
- Make the source code available online and ready to install. If possible, use a simulation platform to implement the model, hence fostering software reuse in order to make simulations reliable and more comparable to each other. If possible, make the simulation available to be run online, for instance, by using such technologies as Applets, a technology that allows the embedding of an execution model in Web pages, making it possible to be loaded and executed remotely. Making the computerised model available is crucial for others to be able to run the model with parameters settings that were not reported in your papers. Whereas for a certain set of parameter settings two simulations may match, this may not happen with other settings, suggesting the two models are not equivalent.
- Provide a detailed description about the results, statistical methods used, distributional information and qualitative measures. Make the bulk outputs available online or in appendices.

While programming for replicability is something to be considered on the part of the team that programs the model, a number of aspects should be considered by the team that decides to replicate a model. Often apparently irrelevant differences in two implementations can be the cause of different statistics that do not match (Edmonds and Hales 2003). This is particularly relevant if the description of the original model is not detailed sufficiently, making it difficult to assess whether the computerised model is implemented correctly and according to the conceptual model. When the original implementation is available, high degrees of confidence in the new implementation requires matching results with runs based on different parameters from those reported in the original model. The experiment of Edmonds and Hales provides a good deal of informative techniques and tips on this kind of model alignment:

- If the simulation shows very dynamic self-reinforcing effects in the long run, check the alignment of simulations in the first few cycles of their runs, averaged over several runs, in order to test whether they are initialized in the same way.
- Use different parameter settings and for each setting make several runs of both implementations over long-term periods. For each implementation collect several statistics from each time period and average them over the number of runs. For each such pair of sets of averages use statistical tests, such as the Kolmogorov-Smirnov test for the goodness of fit of cumulative distribution functions.
- Use modularity to test features separately. If two simulations do not align, turn off certain features of the implementations until they do so, and find the source of errors in the different modules. Reprogramming whole critical modules may also apply here.
- Use different toolkits or programming languages to re-implement simulations and if possible have this done by different people.

8.3.5 Participative-Based Methods

Replication is feasible when models are simple. When the goal is modelling a specific target domain, full of context, with significant amounts of rich detail, and use of empirical data and stakeholder participation, such as with the Companion Modelling approach, replication may not be feasible for verifying the computerised model. As in any other simulation, good programming practices and defensive programming are thus fundamental. In addition, insofar as some results may be due to software faults and be unexpected for stakeholders, participative-based methods are also a form of verification. This fact stresses the importance of involving domain experts and stakeholders as much as possible in all stages of the development and implementation process. Here, documentation and visualisation techniques can play a crucial role in bridging between the stakeholders' opinions and the intention of the programmer of the simulation. This is discussed in more detail in Chap. 10 in this volume (Barreteau et al. 2013).

8.4 Validation Approaches

We offered a conceptual definition of validation in Sect. 8.2.2. Had we given an operational definition, things would have become somewhat problematical. Models of social complexity are diverse and there is no definitive and guaranteed criterion of validity. As Amblard et al. (2007) remarked, "validation suggests a reflection on the intended use of the model in order to be valid, and the interpretation of the results should be done in relation to that specific context."

A specific use may be associated with different methodological perspectives for building the model, with different strategies, types of validity tests, and techniques - Fig. 8.6. Consider the kind of subjunctive, metaphorical, models such as Schelling's. In these models there is no salient validation step during the simulation development process. Design and validation walk together, and the intended use is not to show that the simulation is plausible against a specific context of social reality but to propose abstract or schematic mechanisms as broad representations of classes of social phenomena. In other different cases, the goal may be modelling a specific target domain, full of context, with use of empirical data and significant amounts of rich detail. Whereas in the former case a good practice could be modelling with the greatest parsimony possible so as to have a computational model sanctionable by human beings and comparable to other models, parsimony can be in opposition to the goal of descriptive richness and thus inappropriate to the latter case.

There are also different methodological motivations behind the use of a model, such as those conceived to predict or explain and those merely conceived to describe. Regardless of what method is used, the reproduction of characteristics of the object domain is important, but this can be assessed through rather different approaches during the model development process. If it is prediction you are

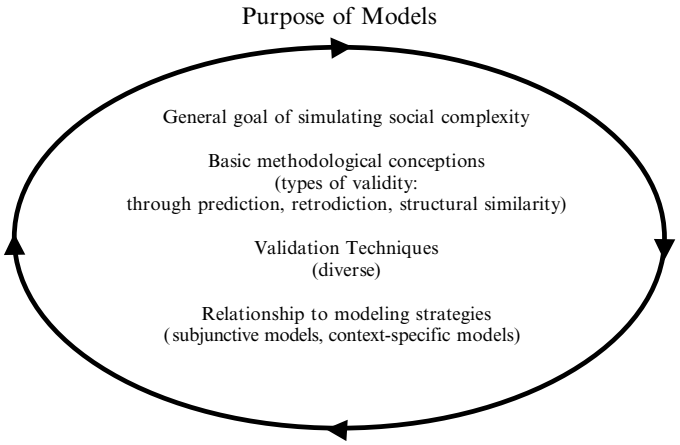

Fig. 8.6 Validation implies considering the purpose of the model

seeking, validation consists of confronting simulated behaviour with the future behaviour of the target system (however, attempting to establish numerical prediction is not a normal goal in simulation). If it is explanation, validation consists of building plausible mechanisms that are able to reproduce simulated behaviour similar to real behaviour. If the goal is the more general aim of descriptiveness, explanation may probably be a goal as well, and a creative integration of ways for assessing the structure and results of the model, from quantitative to qualitative and participatory approaches, will be applied.

In conclusion, one should bear in mind that there is no one special method for validating a model. However, it is important to assess whether the simulation is subjected to good practices during its conception, whether it fits the intended use of the model builder and whether it is able to reproduce characteristics of the object domain. Assessing whether the goals of the modellers are well stated and the models themselves are well described in order to be understood and sanctioned by other model builders are *sine qua non* conditions for good simulation modelling.

In the remainder of this section, we revise the issue of the purpose of validating simulations along four dimensions: the general goal of simulation in social complexity, three basic methodological conceptions of validity types, a set of usual techniques applied in social simulation, and finally the relationship of validation to different modelling strategies with respect to the level of descriptive detail embedded in a simulation.

8.4.1 The Goal of Validation: Goodness of Description

If one is using a predictive model, then the purpose of the model is to predict either past or future states of the target system. On the other hand, one may strive for a

model that is able to describe the target system with satisfactory accuracy in order to become more knowledgeable about the functioning of the system, to exercise future and past scenarios, and to explore alternative designs or inform policies.

The objective in this section is to define the purpose of validation in terms of the purpose of simulating social complexity, which we will define as being of good description. This position entails that there is no single method or technique for validating a simulation. A diversity of methods for validating models is generally applied.

In the rest of this chapter we adopt the multi-agent paradigm for modelling. A conceptual understanding of validation, similar but more general than Moss and Edmonds' (2005), will be used:

> The purpose of validation is to assess whether the design of micro-level mechanisms, put forward as theories of social complexity validated to arbitrary levels, can be demonstrated to represent aspects of social behaviour and interaction that are able to produce macro-level effects either (i) broadly consistent with the subjacent theories; and/or (ii) qualitatively or quantitatively similar to real data.

By broad consistency we mean the plausibility of both micro specification and macro effects accounted as general representations of the target social reality. In its most extreme expression, plausibility may be evaluated on a metaphorical basis. By qualitative similarity to real data we mean a comparison with the model in terms of categorical outcomes, accounted as qualitative features, such as the shape of the outcomes, general stylized facts, or dynamical regimes. As for quantitative similarity we mean the very unlikely case in which the identification of formal numerical relationships between aggregate variables in the model and in the target – such as enquiring as to whether both series may draw from the same statistical distribution – proves to be possible.

Notice that this definition is general enough to consider both the micro-level mechanisms and macro-level effects assessed on a participatory basis. It is also general enough to consider two methodological practices – not necessarily incompatible – related to the extent to which models in social science simulation ought to be constructed on the basis of formal theories or ought to be based on techniques and approaches on the basis of the intuition of the model builders and stakeholders – an issue that we will come back to later. These are omnipresent methodological questions in the social simulation literature and are by no means irrelevant to the purpose of simulation models.

Suppose that on the basis of a very abstract model, such as the Schelling model, you were to evaluate the similarity of its outputs with empirical data. Then you will probably not take issue with the fact that the goal of predicting future states of the target would be out of the scope of simulation research for that kind of modelling. However, despite the belief that other sorts of validation are needed, this does not imply excluding the role of prediction, but emphasises the importance of description as the goal of simulating social complexity. In truth, what could be more contentious in assessing the Schelling model is the extreme simplicity used to describe the domain of social segregation. The descriptive power of multi-agent models makes them suited to model social complexity. Computational modelling corresponds to a

process of abstraction, in that it selects some aspects of a subject being modelled, like entities, relations between entities and change of state, while ignoring those that may be considered less relevant to the questions that are of interest to the model builder. The expressiveness of multi-agent models allows the researcher to play with intuitive representations of different aspects of the target, such as defining societies with different kinds of agents, organizations, networks and environments, which interact with each other and represent social heterogeneity. By selecting certain aspects of social reality into a model, this process of demarcation makes multi-agent modelling suited to represent sociality as perceived by researchers and often by the stakeholders themselves.

The descriptive power of simulation is on par with the diversity of ways used for informing the construction and validation of models, from theoretic approaches to the use of empirical data or stakeholder involvement. At any rate, measuring the goodness of fit between the model and real data expressed with data series is neither the unique nor a typical criterion for sanctioning a model. The very idea of using a diversity of formal and informal methods is to assess the credibility of the mechanisms of the model as good descriptions of social behaviour and interaction, which must be shown to be resilient in the face of multiple tests and methods, in order to provide robust knowledge claims and allow the model to be open to scrutiny.

8.4.2 Broad Types of Validity

When we speak about types of validity we mean three general methodological perspectives for assessing whether a model is able to reproduce expected characteristics of an object domain: validation through prediction, validation through retrodiction and validation through structural similarity. Prediction refers to validating a model by comparing the states of a model with future observations of the target system; retrodiction compares the states of the model with past observations of the target system; and structural similarity refers to assessing the realism of the structure of the model in terms of empirical and/or theoretical knowledge of the target system. In practice, all three approaches are interdependent and no single approach is used alone.

8.4.2.1 Validation Through Prediction

Validation through prediction requires matching the model with aspects of the target system before they were observed. The logic of predictive validity is the following: If one is using a *predictive model* – in which the purpose of the model is to predict future states of the target system – and the predictions prove satisfactory in repeated tested events, it may be reasonable to expect the model outcomes to stay reliable under similar conditions (Gross and Strand 2000). The purpose of prediction is somewhat problematic in social simulation:

- Models of social complexity usually show nonlinear effects in which the global behaviour of the model can become path-dependent and self-reinforcing, producing high sensitivity to initial conditions, which limits the use of predictive approaches.
- Many social systems show high volatility with unpredictable events, such as turning points of macroeconomic trade cycles or of financial markets that are in practice (and possibly in principle) impossible to predict; see (Moss and Edmonds 2005) for a discussion on this.
- Many social systems are not amenable to direct observation, change too slowly, and/or do not provide enough data to be able to compare model outcomes. Most involve human beings and are too valuable to allow repeated intervention, which hinders the acquisition of knowledge about its future behaviour. Policies based on false predictions could have serious consequences, thus making the purpose of prediction unusable (Gross and Strand 2000).

While quantitative prediction of the target system behaviour is rare or simply unattainable, prediction in general is not able to validate per se the *mechanisms* of the model as good representations of the target system. In the words of Troitzsch (2004), "What simulations are useful to predict is only how a target system might behave in the future qualitatively". But a different model using different mechanisms that could lead to the same qualitative prediction may always exist, thus providing a different explanation for the same prediction. More often, the role of predicting future states of the target system becomes the exploration of new patterns of behaviour that were not identified before in the target system, whereby simulation acquires a speculative character useful as a heuristic and learning tool. What we are predicting is really new concepts that we had not realized as being relevant just for looking into the target.

8.4.2.2 Validation Through Retrodiction

The difference from retrodiction to prediction is that in the former the intention is to reproduce *already* observed aspects of the target system. Given the existence of a historical record of facts from the target system, the rationale of *retrodictive* validity for a *predictive* model is the following: If the model is able to reproduce a historical record consistently and correctly, then the model may also be trusted for the future (Gross and Strand 2000). However, as we have mentioned, predictive models of social complexity are uncommon in simulation. Explanation rather than prediction is the usual motive for retrodiction. The logic of retrodictive validity is the following: If a model is able to consistently reproduce a record of past behaviours of the target system, then the mechanisms that constitute the model are *eligible candidates* for explaining the functioning of the target system. Nevertheless, retrodiction alone is not sufficient to assess the validity of the candidate explanations:

- Underdetermination: Given a model able to explain a certain record of behaviours or historical data, there will always be a different model yielding a different explanation for the same record.
- Insufficient quality of data: In many cases it is impossible to obtain long historical series of social facts in the target system. In the social sciences the very notion of social facts or data is controversial, can be subjective, and is not dissociable from effects introduced by the measurement process. Moreover, even when data is available it may not be in a form suitable to be matched to the bulk of data generated by simulation models.

Underdetermination and insufficient data suggest the crucial importance of domain experts for validating the *mechanisms* specified in the model. A model is only valid provided that *both* the generated outcomes and the mechanisms that constitute the model are sanctioned by experts in the relevant domain. The importance of validating the mechanisms themselves leads us to the *structural* validity of the model, which neither predictive nor retrodictive validity is able to assess alone.

8.4.2.3 Validation Through Structural Similarity

In practice, the evaluation of a simulation includes some kind of prediction and retrodiction, based on expertise and experience. Given the implementation of micro-level mechanisms in the simulation, classes of behaviour at the macroscopic scale are identified in the model and compared to classes of behaviour identified in the target. Similarly, known classes of behaviour in the target system are checked for existence in the simulation. The former case is generally what we call the "surprising" character of simulations in which models show something beyond what we expect them to. However, only an assessment of the model at various points of view, including its structure and properties on different grains and levels, will truly determine whether the system reflects the way in which the target system operates. For instance, do agents' behaviour, the constituent parts and the structural evolution of the model match the conception we have about the target system with satisfactory accuracy? These are examples of the elements of realism between the model and the system that the researcher strives to find, which requires expertise in the domain on the part of the person who builds and/or validates the model.

8.4.3 Validation Techniques

In this section we describe validation techniques used in social simulation. Some are used as common practices in the literature, and most of the terminology has been inhered from simulation in engineering and computer science, particularly from the reviews of validation and verification in engineering by Sargent (1999). All techniques that we describe can be found in the literature, but it would be rare to

find a model in which only one technique was used, consistent with the fact that the validation process should be diverse. Also, there are no standard names in the literature and some techniques overlap with others.

8.4.3.1 Face Validity

Face validity is a general kind of test used both before and after the model is put to use. During the model development process, the various intermediate models are presented to persons who are knowledgeable about, or are relevant to the problem in order to assess whether it is compatible with their knowledge and experience and reasonable for its purpose (Sargent 1999). Face validity may be used for evaluating the conceptual model, the components thereof, and the behaviour of the computerised models in terms of categorical outcomes or direct input/output relationships. This can be accomplished via documentation, graphing visualisation models, and animation of the model as it moves through time. Insofar as this is a general kind of test, it is used in several iterations of the model.

8.4.3.2 Turing Tests

People who are knowledgeable about the behaviour of the target system are asked if they can discriminate between system and model outputs (Sargent 1999). The logic of Turing tests is the following: If the outputs of a computerised model are qualitatively or quantitatively indistinguishable from the observation of the target system, a substantial level of validation has been achieved.

Note that the behaviour of the target system does not need to be observed directly in the cases where a computerised representation is available. For example, suppose that videos of car traffic are transformed into three-dimensional scenes, whereby each object in the scene represents a car following the observed trajectory. If an independent investigator is not able to distinguish the computerised reproduction from an agent-based simulation of car traffic, then a substantial level of validation has been obtained for the set of behaviours represented in the simulation model.

8.4.3.3 Historical Validity

Historical validity is a kind of retrodiction where the results of the model are compared with the results of previously collected data. If only a portion of the available historical data is used to design the model then a related concept is called *out-of-sample tests* in which the remaining data are used to test the predicative capacity of the model.

8.4.3.4 Event Validity

Event validity compares the occurrence of particular events in the model with the occurrence of events in the source data. This can be assessed at the level of individual trajectories of agents or at any aggregate level. Events are situations that should occur according to pre-specified conditions, although not necessarily predictable. Some events may occur at unpredictable points in time or circumstances. For instance, if the target system data shows arbitrary periods of stable behaviours interwoven with periods of volatility with unpredictable turning points, the simulation should produce similar kinds of unpredictable turning events.

8.4.3.5 Extreme Condition Tests

Extreme conditions are used for both verifying and validating the validation. The experimenter uses unlikely combinations of factors in the system, usually very high or low values for the inputs and parameters in order to test whether the simulation continues to make sense at the margins. For instance, if interaction among agents is nearly suppressed the modeller should be surprised if such activities as trade or culture dissemination continues in a population.

8.4.3.6 Sensitivity Analysis

As a precautious rule one should consider that a model is only valid for the range of parameters that have been tested. Sensitivity analysis stands for tests in which parameters or even the inter-relations of model components are systematically varied in order to determine the effect on the behaviour of the model. It aims at three sorts of related considerations:

- Understanding the basic conditions under which the model behaves as expected;
- Finding the conditions that maximize the agreement of the model behaviour with the target system behaviour;
- Identifying the conditions that are sensitive, for instance, when changes in the input yield outputs that do not remain within known intervals, even when changes are carried out in a very controlled way.

Parameters that are sensitive should be made sufficiently accurate prior to using the model. If the output remains unpredictable even with controlled changes, the modeller should be concerned about making claims about the model.

Executing sensitivity tests is not a trivial task, and there are no special methods. If one imagines sweeping three parameters with 10 distinct values each then 720 configurations of input sets will be defined. If for each configuration we carry out five runs so as to obtain meaningful statistical figures, we can imagine 3,600 runs of the model. Since it is likely that some of the parameters will interact they should be

swept in combinations – a fact which makes sensitivity tests already intractable for only a relatively small number of parameters.

Experience, the availability of empirical data, and the use of sampling techniques are the usual solution. A possible approach is constraining the range of values according to empirical data by ignoring ranges of values that are known from the start to be implausible in the target system. Yet, this might not be possible. The correspondence between parameter ranges in the model and in the target must be somehow known a priori, which requires that the model be subjected to some kind of testing anyway.

Sampling the parameter space is the usual solution. Related techniques in social simulation include learning algorithms for searching the parameter space, such as the Active Nonlinear Test (Miller 1998). Genetic algorithms for exploring the parameters of the model more efficiently are often used.

Besides sweeping parameters, any changes in the conditions of the model should be tested. Two architectural levels in the model must be considered:

1. The conceptual level, which involves changing the internal mechanisms or sub-models that constitute the larger model, such as changing the decision processes of the agents, their learning mechanisms or their interaction topology.
2. The system level, which involves low-level elements of the model, such as the effect of changing the agent activation regimes (e.g. uniform activation or random activation).

If changing elements at the system level determines different behaviours of the model that cannot be adequately interpreted, then the validity of the model can be compromised. The case of changing elements at conceptual levels is more subtle, and the validity of the results must be assessed by the researcher with reference to the validity of the composing elements of the model. This is basically a kind of cross-model or cross-element validation, as described below.

8.4.3.7 Cross-Sectional Validity

Cross-sectional validity refers to the examination of the fit of congruence of social data to the results that simulation models produce in a specific point in time. This may be accomplished by comparing empirical data, such as a cross-sectional survey, with output generated by a model at a single time period. For example, simulation models of withdrawal behaviours in organisations, based on fuzzy set theory, have been used by Munson and Hulin (2000) for comparing correlations among frequencies of withdrawal behaviours in a cross-section survey with correlations among simulated behaviours after a certain number of iterations. The model that generated the correlation matrix that fitted better with the empirical data (evaluated by calculating root mean squared discrepancies) gained support as the most useful model of the organisational context that was being analysed. Notwithstanding, quantitative assessments between cross-sectional and simulated data are

rare. Moreover, the benefits of simulation for representing time suggest an obvious disadvantage of cross-sectional approaches: they are unable to assess results from a longitudinal point of view.

8.4.3.8 Comparison to Other Models

There are strong reasons to compare a model with other models. One reason is the unavailability, or insufficient quality, of data. Another is that models are often specified at a level of abstraction not compatible with available data. Moreover, even if data were available, the goodness of fit between real and simulated data, albeit reflecting evidence about the validity of the model as a data-generating process, does not provide evidence on how it operates. The most important reason for comparing models is intrinsic to the scientific practice. The very idea of validation is comparing models with other descriptions of the problem modelled, and this may include other simulation models that have been validated to some level.

If two models of the same puzzle lead to different conclusions, then they motivate us to check the validity of the models. Or, as we mentioned in Sect. 8.2, indicate that the computerised models have not been appropriately verified. The bricolage of methods for relating models in social simulation has become more mature after the aligning experience of Axtell et al. (1996), and is now a highly cited area. The goal of relating models can have different origins. In Sect. 8.2.4 the meaning of *model alignment* was described as well as the different methods for replicating (computerised) models. These focused on the *verification* of algorithmic equivalence of models through comparison of data. There are a number of approaches for relating models focused on the *validation* perspective:

- Extending models or composing models in a larger model, where different concepts, mechanisms or results are abstracted as components of the larger model; software reuse, as described in Sect. 8.2.2, is a kind of model composition.
- Docking data produced by a model to a set of data produced by another model; this may require changing and calibrating the model, for instance, by turning off features of the former in order to align with the latter.
- Varying systematically and controllably the internal structure of a model; in other words, playing with models within models in order to assess the overall validity of the larger model with reference to the validity of each one of the composing models; this is a kind of sensitivity analysis that resembles *cross-element validation*, which is mentioned below.
- Relating different computerised models that are based on different paradigms and provide different scales and perspectives for the same class of target; for instance, agent-based models and equation-based modelling.

8.4.3.9 Cross-Element Validity

Cross-element validation, rather than comparing whole models, compares the results of a model whose architecture of the agents differs only in a few elements. The goal is to assess the extent to which changing elements of the model architecture produces results compatible with the expected results of the (larger) model; basically an exercise of composing different models within a larger model, which resembles structural sensitivity analysis. For instance, one may study the effects of using a model with agents in a bargaining game employing either evolutionary learning or reinforcement learning strategies, and assess which one of the strategies produces results compatible with theoretical analysis in game theory (Takadama et al. 2003).

A difficult issue faced with cross-element validation is that different results depend on the different elements used. The results obtained may only be a result of sensitivity to the elements. How can different tendencies be compared resulting from using different elements in the model? Arai and Watanabe (2008) have introduced a promising approach for use with time-series data. A quantitative comparison method based on the Fourier transform is used for measuring the distance between the results of two models that use different elements (e.g. different learning mechanisms).

Another problem with abstract simulations, in which the context of the empirical referent is vague, refers to the lack of real data for comparing which of the simulation outcomes are more realistic. As a result, the outcomes may only be assessed against a reference theoretical model. In contrast, when enough data from the target system are available, cross-element validation becomes a particular case of validation through retrodiction. For example, the cross-sectional validity tests employed by Munson and Hulin (2000) are a kind of cross-element validation through retrodiction, in which different theoretical models of individual withdrawal behaviours were tested within the same organisational model, with post-model data fit evaluation. Data generated for each theoretical model was compared with data from structured interviews in order to determine which theoretical model provided the best fit to the empirical data.

8.4.3.10 Participatory Approaches for Validation

Participatory approaches refer to the involvement of stakeholders both in the design and the validation of a model. Such an approach, also known as Companion Modelling (Barreteau et al. 2001), assumes that model development must be itself considered in the process of social intervention, where dialogue among stakeholders, including both informal and theoretical knowledge, is embedded in the model development process. Rather than just considering the final shape of the model, both the process and the model become instruments for negotiation and decision making. It is particularly suited for policy or strategy development. This topic is discussed in Chap. 10 “Participatory Approaches” (Barreteau et al. 2013).

8.4.4 Relationship to Modelling Strategies

Regarding the diversity of methodological conceptions for modelling, different strategies may be adopted relating to the level of descriptive richness embedded in the simulations.

Several taxonomies of modelling strategies have been described in the literature (see David et al., 2004; Boero et al., 2005; Gibert 2008, pp. 42–44). Normally, the adoption of these strategies does not depend on the class of the target to be modeled, but in different ways to address it as the problem domain. For example, if a simulation is intended to model a system for the purpose of designing policies, this will imply representing more detail than a simulation intended for modeling certain social mechanisms of the system in a metaphorical way. But this also means that there is a trade-off between the effort that a modeler puts into verifying the simulation and puts into validating it. As more context and richness are embedded in a model, the more difficult it will be to verify it. Conversely, as one increases the descriptive richness of a simulation, more ways will be available to assess its validity. A tension that contrasts the tendency for constraining simulations by formal-theoretical constructs – normally easier to verify – and constraining simulations by theoretical-empirical descriptions – more amenable to validation by empirical and participative-based methods. In the remaining sections of this chapter, two contrasting modelling strategies are described.

8.4.4.1 Subjunctive Agent-Based Models

A popular strategy in social simulation consists of using models as a means for expressing subjunctive moods to talk about possible worlds using what-if scenarios, like 'what would happen if something were the case'. The goal is building artificial societies for modelling possible worlds that represent classes of social mechanisms, while striving for maximal simplicity and strong generalisation power of the representations used. Reasons for striving for simplicity include the computational tractability of the model and to keep the data analysis as simple as possible.

Simplicity and generalization power are often seen as elements of elegance in a model. However, making the model simpler in the social sciences does not necessarily make the model more general. More often than not this kind of modelling only makes it metaphorically general, or simply counterfactual (with false assumptions). For example, 'What would happen if world geography is regarded as a two-dimensional space arranged on a 10×10 grid, where agents are thought of as independent political units, such as nations, which have specific behaviours of interaction according to simple rules?' To assume that world geography is one-dimensional, as Axelrod does in his Tribute Model, is clearly a false assumption. Often these models are associated with a design slogan coined by Axelrod, called the KISS approach – "Keep it Simple Stupid". Despite their simplicity, these models prove useful for concept formation and theoretical abstraction. The

emergence of macro regularities from micro-levels of interaction becomes the source of concept formation and hypothesis illustration, with the power of suggesting novel theoretical debates.

Given the preference for simplification and abstraction, mechanisms used in these models are normally described in a formalized or mathematical way. Axelrod's models, such as the culture dissemination model, or Schelling's residential segregation model, are canonical examples. Their simplicity and elegance have been factors for popularity and dissemination that span numerous disciplines and ease replication and verification.

However, whereas simplicity eases verification, the use of metaphorical models also brings disadvantages. Consider a word composed of several attributes that represents an agent's culture, such as in Axelrod's culture dissemination model. The attributes do not have any specific meaning and are only distinguishable by their relative position in the word and so can be interpreted according to a relatively arbitrary number of situations or social contexts. However, such a representation may also be considered too simplified to mean anything relevant for such a complex concept as a cultural attribute. As a consequence, verification is hardly distinguishable from validation, insofar as the model does not represent a specific context of social reality. In such a sense, the researcher is essentially verifying experimentally whether the conceptions that he has in his mind are met by an operationalisation that is computationally expressed (David et al. 2007). Nevertheless, given their simplicity, subjunctive models can be easily linked and compared to other models, extended with additional mechanisms, as well as modified for model alignment, docking, or replication. Cross-element validation is a widely used technique.

At any rate, the fact that these models are easily replicable and comparable – but hardly falsifiable by empirically acquired characteristics of social reality – stresses their strong characteristic: when models based on strategies of maximal simplicity become accepted by a scientific community, their influence seems to reach several other disciplines and contexts. Perhaps for this reason, these kind of models are the most popular in social simulation, and some models are able to reach a considerable impact in many strains of social science.

8.4.4.2 Context-Specific Agent-Based Models

It would be simplistic to say that models in social simulation can be characterized according to well-defined categories of validation strategies. Even so, the capacity to describe social complexity, whether through simplicity or through rich detail and context, is a determining factor for a catalogue of modelling strategies.

We cannot hope to model general social mechanisms that are valid in all contexts. There are many models that are not designed to be markedly general or metaphorically general, but to stress accurateness, diversity, and richness of description. Instead of possible worlds representing very arbitrary contexts, models are explicitly bounded to specific contexts. Constraints imposed on these models can vary from models investigating properties of social mechanisms in a large band

of situations which share common characteristics, to models with the only ambition of representing a single history, like Dean's retrodiction of the patterns of settlement of a specific population in the southwestern United States, household by household (see Dean et al. 2000).

Constructing and validating a model of this kind requires the use of empirical knowledge. They are, for this reason, often associated with the idea of "Empirical Validation of Agent-Based Models."

What is the meaning of empirical in this sense? If the goal is to discuss empirical claims, then models should attempt to capture empirically enquired characteristics of the target domain. Specifying the context of descriptions in the model will typically provide more ways for enquiring quantitative and qualitative data in the target, as well as using experimental and participative methods with stakeholders. In this sense, empirical may be understood as a stronger link between the model and a context-specific, well-circumscribed, problem domain.

The model of Dean et al. (2000), which attempted to retrodict the patterns of settlement of the Anasazi in the Southwestern United States, household by household, is a well-known and oft-cited example of a highly contextualized model built on the basis of numerous sources, from archaeological data to anthropological, agricultural and ethnographic analyses, in a multidisciplinary context.

Given the higher specificity of the target domain, the higher diversity of ways for enriching the model as well as the increased semantic specificity of the outputs produced by the model, context-specific models may be more susceptible to be compared with empirical results of other methods of social research. On the other hand, comparison with other simulation models is complex and these models are more difficult to replicate and verify.

8.4.4.3 Modus Operandi: Formal and Informal Approaches

The tension between simplicity and descriptive richness expresses two different ways for approaching the construction and validation of a model. One can start with a rich, complex, realistic description and only simplify it where this turns out to be possible and irrelevant to the target system – known as the KIDS approach (Edmonds and Moss 2005). Or one starts from the outset with the simplest possible description and complexifies it only when it turns out to be necessary to make the model more realistic, nevertheless keeping the model as simple as possible – known as the KISS approach (Axelrod 1997b).

In practice, both trends are used for balancing trades-offs between the model's descriptive accuracy and the practicality of modelling, according to the purpose and the context of the model. This raises yet another methodological question: the extent to which models ought to be designed on the basis of formal theories, or ought to be constrained by techniques and approaches just on the basis of the intuition of the model builders and stakeholders. As we have seen, strong, subjunctive, agent-based models with metaphorical purposes tend to adopt the simplicity motto with extensive use of formal constructs, making the models more elegant

from a mathematical point of view, easier to verify, but less liable to validation methods. Game theoretical models, with all their formal and theoretical apparatus, are a canonical example. Results from these models are strongly constrained by the formal theoretical framework used.

A similar problem is found when agent-based models make use of cognitive architectures strongly constrained by logic-based formalisms, such as the kind of formalisms used to specify BDI-type architectures. If the cognitive machinery of the agents relies on heuristic approaches that have been claimed valid, many researchers in the literature claim that cognitive agent-based models can be validated in the empirical sense of context-specific models. Cited examples of this kind usually point to agent-based models based on the Soar architecture.

At any rate, context-specific models are normally more eclectic and make use of both formal and informal knowledge, often including stakeholder evidence in order to build and validate the models. Model design tends to be less constrained a priori by formal constructs. In principle, one starts with all aspects of the target domain that are assumed to be relevant and then explores the behaviour of the model in order to find out whether there are aspects that do not prove relevant for the purpose of the model. The typical approach for modelling and validation can be summarized in a cycle with the following iterative and overlapping steps:

A. *Building and validating pre-computerised and computerised models*: Several descriptions and specifications are used to build a model, eventually in the form of a computer program, which are micro-validated against a theoretical framework and/or empirical knowledge, usually qualitatively. This may include the individual agents' interaction mechanisms (rules of behaviour for agents or organisations of agents), their internal mechanisms (e.g. their cognitive machinery), the kind of interaction topology or environment, and the passive entities with which the agents interact. The model used should be as accurate as possible for the context in consideration as well as flexible for testing how parameters vary in particular circumstances. Empirical data – if available – should be used to help configure the parameters. Both the descriptions of the model and the parameters used should be validated for the specific context of the model. For example, suppose empirical data are available for specifying the consumer demand of products. If the demand varies from sector to sector, one may use data to inform the distribution upon which the parameter could be based for each specific sector.
B. *Specifying expected behaviours of the computerised model*: Micro and macro characteristics that the model is designed to reproduce are established from the outset based on theoretical and/or empirical knowledge. Any property, from quantitative to qualitative measures, such as emergent key facts the model should reproduce (stylized facts), the statistical characteristic or shape of time-data series (statistical signatures) and individual agents' behaviour along the simulation (individual trajectories), can be assessed. This may be carried out in innumerable ways, according to different levels of description or grain, and be more or less general depending on the context of the model and the kind of empirical

knowledge available. For instance, in some systems it may be enough to predict just a "weak" or "positive" measure on some particular output, such as a positive and weak autocorrelation. Or we might look for the emergence of unpredictable events, such as stable regimes interleaved with periods of strong volatility, and check their statistical properties for various levels of granularity. Or the emergence of different structures or patterns associated with particular kinds of agents, such as groups of political agents with "extremist" or "moderate" agents.

C. *Testing the computerised model and building and validating post-computerised models*: The computerised model is executed. Both individual and aggregate characteristics are computed and tested for sensitivity analysis. These are micro-validated and macro-validated against the expected characteristics of the model established in step B according to a variety of validation techniques, as described in Sects. 8.4.2 and 8.4.3. A whole process of building post-computerised models takes place, possibly leading to the discovery of unexpected characteristics in the behaviour of the computerised model which should be assessed with further theoretical or empirical knowledge about the problem domain.

Further Reading

Good introductions to validation and verification of simulation models in general are Sargent (1999) and Troitzsch (2004), the latter with a focus on social simulation. Validation of agent-based models in particular is addressed by Amblard and colleagues (Amblard et al. 2007).

For readers more interested in single aspects of V&V with regard to agent-based models in the context of social simulation, the following papers provide highly accessible starting points:

- Edmonds and Hales (2003) demonstrate the importance of model replication (or model alignment) by means of a clear example.
- Boero and Squazzoni (2005) examine the use of empirical data for model calibration and validation and argue that "the characteristics of the empirical target" influence the choice of validation strategies.
- Moss and Edmonds (2005) discuss an approach for cross-validation that combines the involvement of stakeholders to validate the model qualitatively on the micro level with the application of statistical measures to numerical outputs to validate the model quantitatively on the macro level.

Finally, for a more in-depth epistemological perspective on verification and validation I would refer the inclined reader to a revised version of my EPOS 2006 paper (David 2009).

References

Amblard F, Bommel P, Rouchier J (2007) Assessment and validation of multi-agent models. In: Phan D, Amblard F (eds) Agent-based modelling and simulation in the social and human sciences. The Bardwell Press, Oxford, pp 93–114

Arai R, Watanabe S (2008) A quantitative comparison method for multi-agent based simulation in the feature space. In: David N, Sichman JS (eds) Multi-agent-based simulation IX, MABS 2008, revised selected papers (Lecture notes in artificial intelligence), vol 5269. Springer, Berlin, pp 154–166

Axelrod R (1997a) The dissemination of culture: a model with local convergence and global polarization. J Conf Resolut 41(2):203–226

Axelrod R (1997b) Advancing the art of simulation in the social sciences. In: Conte R, Hegselmann R, Terna P (eds) Simulating social phenomena. Springer, Berlin, pp 21–40

Axtell R, Axelrod R, Epstein J, Cohen M (1996) Aligning simulation models: a case study and results. Comp Math Organiz Theory 1(2):123–141

Barreteau O, Bousquet F, Attonaty JM (2001) Role-playing games for opening the black box of multi-agent systems: method and lessons of its application to Senegal River Valley irrigated systems. J Artif Soc Soc Simul 4(2). http://jasss.soc.surrey.ac.uk/4/2/5.html

Barreteau O et al (2013) Participatory approaches. Chapter 10 in this volume

Boero R, Squazzoni F (2005) Does empirical embeddedness matter? Methodological issues on agent-based models for analytical social science. J Artif Soc Soc Simul 8(4). http://jasss.soc.surrey.ac.uk/8/4/6.html

David N (2009) Validation and verification in social simulation: patterns and clarification of terminology. In: Squazzoni F (ed) Epistemological aspects of computer simulation in the social sciences, second international workshop, EPOS 2006, Brescia, 5–6 Oct 2006, revised selected and invited papers (Lecture notes in computer science), vol 5466. Springer, Berlin, pp 117–129

David N, Sichman J, Coelho H (2003) Towards an emergence-driven software process for agent-based-simulation. In: Sichman JS, Bousquet F, Davidsson P (eds) Multi-agent-based simulation II, third international workshop, MABS 2002, Bologna, 15–16 July 2002, revised papers (Lecture notes in artificial intelligence), vol 2581. Springer, Berlin, pp 89–104

David N, Marietto M, Sichman J, Coelho H (2004) The structure and logic of interdisciplinary research in agent-based social simulation. J Artif Soc Soc Simul 7(3). http://jasss.soc.surrey.ac.uk/7/3/4.html

David N, Sichman J, Coelho H (2007) Simulation as formal and generative social science: the very idea. In: Gershenson C, Aerts D, Edmonds B (eds) Worldviews, science and us: philosophy and complexity. World Scientific, Singapore, pp 266–284

David N, Caldas JC, Coelho H (2010) Epistemological perspectives on simulation III: selected papers of EPOS 2008. J Artif Soc Soc Simul 13(1), special section. http://jasss.soc.surrey.ac.uk/13/1/

Dean S et al (2000) Understanding Anasazi culture change through agent-based modeling. In: Kohler T, Gumerman G (eds) Dynamics in human and primate societies: agent-based modeling of social and spatial processes. Oxford University Press, New York/Oxford, pp 179–205

Edmonds B, Hales D (2003) Replication, replication and replication: some hard lessons from model alignment. J Artif Soc Soc Simul 6(4). http://jasss.soc.surrey.ac.uk/6/4/11.html

Edmonds B, Moss S (2005) From KISS to KIDS: an 'anti-simplistic' modelling approach. In: Davidsson P, Logan B, Takadama K (eds) Multi-agent and multi-agent-based simulation (Lecture notes in artificial intelligence), vol 3415. Springer, Berlin, pp 130–144

Frank U, Troitzsch KG (2005) Epistemological perspectives on simulation. J Artificial Soc Soc Simul 8(4). http://jasss.soc.surrey.ac.uk/8/4/7.html

Galán JM et al (2013) Checking simulations: detecting and avoiding errors and artefacts. Chapter 6 in this volume

Gilbert N (2008) Agent-based models, vol 153, Quantitative applications in the social sciences. Sage, London

Gross D, Strand R (2000) Can agent-based models assist decisions on large-scale practical problems? A philosophical analysis. Complexity 5(6):26–33

Kleijnen JPC (1995) Verification and validation of simulation models. Euro J Operat Res 82: 145–162

Law A, Kelton D (1991) Simulation modelling and analysis, 2nd edn. McGraw Hill, New York

Miller JH (1998) Active nonlinear tests (ANTs) of complex simulation models. Manage Sci 44(6): 820–830

Moss S, Edmonds B (2005) Sociology and simulation: statistical and qualitative cross-validation. Am J Sociol 110(4):1095–1131

Munson L, Hulin C (2000) Examining the fit between empirical data and theoretical simulations. In: Ilgen DR, Hulin CL (eds) Computational modeling of behaviour in organisations: the third scientific discipline. American Psychological Association, Washington, DC, pp 69–84

Rouchier J, Cioffi-Revilla C, Polhill JG, Takadama K (2008) Progress in model-to-model analysis. J Artif Soc Soc Simul 11(2). http://jasss.soc.surrey.ac.uk/11/2/8.html

Sargent R (1999) Verification and validation of simulation models. In: Farrington PA, Nembhard HB, Sturrock DT, Evans GW (eds) Proceedings of the 1999 winter simulation conference, Squaw Peak, Phoenix, 5–8 Dec 1999, pp 39–48. http://www.informs-sim.org/wsc99papers/005.PDF

Sommerville I (1995) Software engineering, 5th edn. Addison Wesley, Boston

Squazzoni F (2009) Epistemological aspects of computer simulation in the social sciences, second international workshop, EPOS 2006, Brescia, Revised selected and invited papers (Lecture notes in computer science), vol 5466. Springer, Berlin, 5–6 Oct 2006

Takadama K, Suematsu Y, Sugimoto N, Nawa N, Shimohara K (2003) Cross-element validation in multi-agent-based simulation: switching learning mechanisms in agents. J Artif Soc Soc Simul 6(4). http://jasss.soc.surrey.ac.uk/6/4/6.html

Troitzsch K (2004) Validating simulation models. In: Horton G (ed) Proceedings of the 18th European simulation multiconference, ESM 2004, 13–16 June 2004, Magdeburg. SCS, Erlangen, pp 265–270