On Generating Hypotheses using Computer Simulations

Kathleen M. Carley
Social and Decision Sciences
and
H.J. Heinz III School of Public Policy and Management
Carnegie Mellon University

Tel: 1-412-268-3225
Fax: 1-412-268-6938
Email: carley+@andrew.cmu.edu
url: http://sds.hss.cmu.edu/faculty/carley/carley.htm

Keywords: simulation, computer modeling, hypotheses, organizations, dynamic systems

Grant Sponsors: ONR, NSF

Contract grant numbers:
ONR N00014-97-1-0037
NSF IRI9633 662

Citation: Kathleen M. Carley, 1999, "On Generating Hypotheses Using Computer Simulations. " *Systems Engineering*, 2(2): 69-77. (short version in CCRTS Proceedings)

Abstract

Computational modeling is increasingly being used to do theory development. Computer simulation, numerical estimation, and emulation models are used to describe complex systems and generate a series of hypotheses about the behavior of these systems under different scenarios. These hypotheses can serve as guides to the design of human laboratory experiments and can suggest what data to collect in field and gaming situations. Models that are particularly useful in this venue are process models of non-linear systems where multiple factors dynamically interact. This chapter describes these models and illustrates how they can be used to generate a series of hypotheses that guide future endeavors.

On Generating Hypotheses using Computer Simulations

Kathleen M. Carley

1 Introduction

The use of formal techniques in general, and computational analysis in particular, is playing an ever increasingly important role in the development of theories of complex systems such as groups, teams, organizations, and their command and control architectures. One reason for this is the growing recognition that the underlying processes are complex, dynamic, adaptive, and non-linear, that group or team behavior emerges from interactions within and between the agents and entities that comprise the unit (the people, sub-groups, technologies, etc.), and that the relationships among these entities constrain and enable individual and unit level action [Nowak and Latane, 1994; Carley, 1999]. Another reason for the movement to computational approaches is the recognition that units are inherently computational since they have a need to scan and observe their environment, store facts and programs, communicate among members and with their environment, and transform information by human or automated decision making [Baligh, Burton and Obel, 1990; Carley and Gasser, 1999]. In general, the aim of this computational research is to build new concepts, theories, and knowledge about complex systems such as organizations, groups, teams, or command and control architectures. This aim can be, and is being met, through the use of a wide range of computational models including computer-based simulation, numerical enumeration, and emulation[1] models that focus on the underlying processes.

There are many different types of computational models of process. However, most of them have the following features in common:

1. They are event driven. Thus, they enable the sequences and patterns of events to be analyzed. This facilitates looking at actions, activities, and process. This also makes it possible to set up the timing of actions in these models to correspond to timing in the system being modeled.
2. They allow the agents in the team or group to differ. Thus, rather than modeling the unit as a collective with certain features, each agent in the unit (such as each team member) is modeled as a unique entity. What this means is that the agents in these models can be heterogeneous with respect to key factors such as training, ability to learn, knowledge, interaction partners, and so on. In contrast, many economic models assume actor homogeneity.
3. The computational models are computationally eclectic and include a combination of equations, logic, agent models, and symbolic logic. Models used to simulate processes, particularly organizational processes often employ combined systems. For example, the ORGAHEAD model used herein combines a simulated annealer with a system that operates, more or less, as a neural-net.
4. There are few input parameters relative to the number of possible predictions as much of the action in the model is generated by the processes.
5. The models are applicable to multiple problems. Since the computational models are models of process, they are applicable in a wide number of situations. Unlike comparative statics models with compare two equilibrium situations, these models can be used to explore the process of change. For example, ORGAHEAD can be used to

model small teams, university departments, small companies, joint task forces, etc. However, tuning the model to a new task or situation may be an intensive exercise requiring a high level of expertise about that task or situation. Thus, new subject matter experts may be needed for each new instantiation of the basic model.

6 These computational models are themselves the theory that is being developed and tested.

7 The computational model does the task. Thus, in laboratory experiments, the model, with some modification, can be adjusted to take the place of a human in the experiment.

8 Since the models are models of the underlying process they generate a large number of hypotheses in multiple situations. These hypotheses can then be tested in other settings.

9 Finally, these models are sufficiently complex that they are impossible to completely validate. However, key inputs, processes, and outputs can and should be validated. The type and level of validation should be suited to the model and its purpose [Burton and Obel, 1995].

These nine features dictate the methods of analysis, validation and verification that can be used on these computational models of process. Historically, it was thought that computational process models could be tested and validated using the Turing test. The initial argument was that process models do the task they seek to explain. Thus the model itself is substitutable for the entity that is being modeled. Thus the model itself is the theory. With this in mind the original Turing test was proposed.

The goal behind a Turing test is to see whether or not the observer can distinguish between results generated in two fashions. These results may be generated by two alternative computational models, by a computational model and an analytical model, by a computational model and an experiment, and so on. Turing tests may or may not be rigorous, may or may not employ the use of quantitative data, and may or may not be carried out statistically. Turing tests have typically been employed in simulations of machines or of single humans.

For units, or multi-agent models, the difficulty of the Turing test is that it is incomplete from a social or group perspective. Carley and Newell [1994] suggest that when the computational model is meant to act as a social agent or a group of social agents it is more appropriate to use a revised version of the Turing test which they refer to as the Social Turing Test. Social Turing tests are conducted like Turing tests except that they take into account the group context and expect social not just behavioral responses. To do a social Turing test the following three steps are followed.

Construct a collection of social agents according to the hypotheses and put them in a social situation, as defined by the hypotheses. Recognizably social behavior should emerge from the model.

Aspects of the computational model that are not specified by the hypotheses, of which there will be many, can be determined at will. In general, such aspects should be set based on known human data or handled using Monte Carlo techniques.

The behavior of the computational system can vary widely with such specification, but it should remain recognizably social. The behavior of a model is recognizably social if the behavior of a unit (two or more agents) acts in a way that takes account of social knowledge and the actions and interactions among the agents, cannot be generated from the behavior of a single agent, and if the behavior of the computational model matches, within some pre-defined level of fit, the behavior of a human group in terms of qualitatively or quantitatively fitting known data on a human unit or

fitting a subject matter expert's description of the human unit's behavior. If so, then the Social Turing Test is met.

An increasing large number of claims are being made about the value and use of computer-based simulation in general and computational process models in particular. These claims appear in articles in almost every discipline. One of the strongest claims is that such computer-based simulation can be used for theory development and hypothesis generation. Simple, but non-linear processes, often underlie the team and group behavior [see for example, Carley, 1999; Epstein and Axtell, 1997]. An example of such a non-linearity is the decreasing ability of a new piece of information to alter an agents opinion as the agent gains experience. As the agent gets more and more information that confirms a previously held idea, any information that disconfirms it is increasingly discounted. Such non-linearities make it non-trivial to think through the results of various types of learning, adaptation, and response of teams and groups, particularly in changing environments. Computational analysis enables the theorist to think through the possible ramifications of such non-linear processes and to develop a series of consistent predictions. These predictions are the hypotheses that can then be tested in human laboratory experiments or in live simulations. Thus, computational models can be, and have been, used in a normative fashion to generate a series of hypotheses by running virtual experiments. This approach can be followed whether one is using a symbolic model, multi-agent model, or any other form of computational model.

Another contribution of computational process models is that they can be used to demonstrate gaps in extant verbal theories and the consistency of the predictions made using such theories. There are of course many other claims for, and uses of process models. These include: determine necessity of a posited mechanism, suggest critical experiments, suggest critical items for surveys, suggest relative impact of different input parameters, suggest limits to statistical tests for non-linear systems, substitute for person, group, tool, etc. in an experiment. All of these uses are important from a model driven experimental perspective.

2 Virtual Experiments

One of the most effective ways of generating hypotheses from computational models is by running a virtual experiment. A virtual experiment is an experiment in which the data for each cell in the experimental design is generated by running a computer simulation model. In generating this experiment, standard principles of good experimental design should be followed. The results should then be analyzed statistically. The results of that analysis are the hypotheses that can be examined using data from human laboratory experiments, live simulations, games, field studies, or archival sources. To demonstrate the value of this approach a particular illustrative virtual experiment is described.

3 Illustrative Virtual Experiment

To illustrate how a virtual experiment is done and hypotheses generated, a specific virtual experiment that was run using ORGAHEAD is described [Carley, 1996a; Carley 1998; Carley and Svoboda, 1996; Carley and Lee, 1998]. ORGAHEAD illustrates several aspects of computational process models:

1. ORGAHEAD has been built in a building block fashion by adding on to a base model additional computational process modules. This building block approach is one of the

strongest approaches for building computational models as it enables the designer to validate the model as it is developed and to generate intermediate results.

2. Computational process modules should have face validity. ORGAHEAD, has demonstrated this level of validity and captures the core aspects of unit level architecture.
3. ORGAHEAD, like any computational process model, enables huge numbers of predictions in multiple areas.
4. ORGAHEAD, like any good computational process model, is testable.

In particular the concern in this chapter is with point 3. Thus ORGAHEAD is used to illustrate how virtual experiments are done and hypotheses generated. The goal here is not to describe ORGAHEAD or its predictions in depth. The reader should turn elsewhere for that information [e.g., see Carley and Svoboda, 1996].

For the sake of clarity of exposition, a brief description of ORGAHEAD is provided. ORGAHEAD is a computational process model of organizational performance. It predicts variation in performance for units with different command and control architectures and different task environments. Each agent in the organization is modeled and learns on its own. ORGAHEAD is a dual-level information processing model of strategic adaptation in which the commander can change the organization's form (e.g., the C^2 architecture) in response to various external and internal triggers. At the operational level, each unit is modeled as a set of intelligent adaptive agents (the decision making units - DMU's) arranged in a command structure. Each DMU-agent may be either a person a subgroup, or a platform. Agents are boundedly rational and so exhibit limited attention, memory, information processing capability, and access to information. Unit level performance is determined by the agent's actions as they process tasks. At the strategic level, the commander can alter the C^2 architecture strategically in response to changes in the environment and units actual and expected performance. The types of changes examined herein are retasking and reassignment of personnel. Unit performance is affected by the ability of the commander to anticipate the future and take the appropriate strategic actions to alter the C^2 structure in response to actual or anticipated environmental changes. This strategic adaptation is modeled as having two components an annealing process whereby the commander tunes the extant structure but at a decreasing rate over time, and a shake-up process whereby the commander alters the likelihood of making major changes in the structure (this can be thought of as increased risk taking).

ORGAHEAD can be thought of as a grounded theory of organizational performance as the behavior of the agents and the features of the model have been chosen to reflect findings from empirical studies of individual human learning and organizational adaptation. ORGAHEAD has sufficient versatility that the user can specify the initial C^2 architecture of one or more units, basic training procedures, constraints on agent abilities, the type and likelihood of allowable strategic changes, the maximum frequency of change, the rate of risk aversion, the organization's goal criteria, the task environment, and several types of change "triggers". For example, the C^2 architecture can have from 1 to 4 authority levels with 0 to 15 personnel at each level. All of the agents are boundedly rational but they can vary in their retention level and the number of resources they handle (0 to 7) which includes both those needed for communication and for task analysis.

ORGAHEAD has been used to make predictions about training, learning, the fragility of organizational success, the type of emergent form, the relative value of different organizational forms, etc. Each of these represent a distinct output or observable variable. One of the interesting predictions from ORGAHEAD is that organizations can trade individual experience or learning for

structural learning. Another finding is that, while all successful organizational forms are similar (e.g., employ a similar number of personnel, arrange them into a similar number of sub-groups, have similar levels of hierarchy and span of control, similar workloads, etc.), their nearest neighbor may be a completely unsuccessful form.[2] Thus small changes in an organization's command and control architecture, small changes in a group's structure, can be devastating.

4 Stages in Running a Virtual Experiment

In conducting a virtual experiment and generating a series of hypotheses the followings stages are gone through. (1) Identify key inputs. (2) Explore the input parameter space. (3) Set non-key inputs. (4) Run simulations in virtual experiment. (5) Statistically analyze output. (6) Generate hypotheses.

4.1 *Stage 1: Identify Key Inputs*

Begin by identifying the key inputs. These key inputs, from an experimental design perspective, are the primary or core independent variables. Key inputs are those input parameters of central concern to the theory, or system you are trying to build or understand. These key inputs should be the parameters or model modules which are hypothesized to be the most relevant ones in affecting the output of interest. In general, if the there are input parameters that must be specified to make the computational model operate, but are not ones about which the researcher wishes to generate hypotheses, then those inputs are not considered to be key. These key inputs are the parameters and define the parameter space.

Example: Here the concern is with the dynamic relationship between strategy, structure and performance and their impact on meta-learning. Meta-learning is learning how to learn. The adoption of various change strategies, such as re-assign people rather than fire them, are usually considered to be examples of meta-learning. Using computational analysis we will look for evidence of meta-learning; i.e., evidence that the organizations have evolved effective change strategies. For this example the input parameters are: task limit, task complexity, task information, agent ability, environmental stress, unit size, and shake-up strategies. The outputs include performance, number of re-assignments, and number of re-taskings.

4.2 *Stage 2: Explore the Parameter Space*

Once the input parameters have been identified you need to then define the domain: i.e. what values of each input parameter will be explored. The choice here should reflect concerns with these inputs, and expectations as to where different input values will effect different system level behavior which might alter the output. Sometimes parameter values can be chosen to reflect known values in human groups or teams.

Example: The input parameter space is explored in order to determine the conditions which facilitate meta-learning. Two or more values are chosen for each of the key inputs.

Task limit. The first input parameter, is the simply the number of tasks to which the organization must respond. Since organizations are known to evolve through time and activity, this parameter can be seen as a proxy for time as well, given a naive correspondence between time and activity. We examined organizations under two task limits, 20,000 and 80,000, to determine the differential effects of a longer-term learning period, if any.

Task complexity. The complexity of the task is the size of the solution space for that task. Organizations often face tasks of different complexity. ORGAHEAD allows for several variations of "complexity". In the experiments presented herein, we define a simple task as one which is comprised of signals that take on only two values, yes or no: a "binary task". A complex task is comprised of signals that can take on three values, yes, no, or neutral/maybe; we refer to these as "trinary tasks". As correct solutions become harder to attain, it is expected that the higher performing organizations will show more complex structures than those which are faced with more simple tasks. That is, we should expect a higher degree of meta-learning and, consequently, the complexity of organizational elements, be they relations, individual attributes or meta-adaptive strategies, to increase as the difficulty of the task increases.

Task information. This is the amount of information per task. This is a second aspect of task complexity. We examine tasks where the number of bits of information per task is 7 or 9.

Agent ability. This is the number of incoming signals, or bits of information the agent can consider at one time. Agent ability can be roughly interpreted as how capable is the agent at handling information. We consider two levels of ability low (5) and high (7).

Environmental stress. This refers to whether or not the task environment faced by the organization is consistent. Two domains are considered: no stress and stressful. In the first case, when the input parameter environmental stress is set to no stress then the task environment is stable and does not vary. The tasks to which the unit needs to respond have the same solution criteria throughout the unit's life-cycle as set by the task limit parameter. This is also known as a no-bias environment. In this case, the unit must classify an input vector as "friendly" or "enemy"; the elements of the input vector can be considered as features of a yet to be identified target or assessed situation; and on average half of the features suggest a friendly interpretation. In the second case, when the when the input parameter environmental stress is set to stressful, the task environment periodically oscillates between two different solution criterion. In this case, the task environment oscillates between the no-bias environment just described and a "high alert environment". In the high alert environment the unit is put into a state of alert and the criterion for classifying the target as "friendly" becomes much more stringent. Fewer than half of the features suggest a friendly interpretation. For this situation, we say the environment is in a state of "high bias"; the bias is strongly towards the "enemy" classification. Finally, in the stressful domain the environment oscillates every 5,000 tasks between no-bias and high-bias.

Size. Size is the number of personnel in the unit. While size has always been assumed to impact the unit's behavior and performance, the nature of the effect is disputed. For these experiments, we examine units which are constrained to have maximum numbers of people per level of 3, 4, 6, and 12. These limits correspond to unit sizes of 9, 12, 18, and 36 respectively.

Shake-ups. During a shake-up the unit becomes more risk-taking. For an annealer, such as in ORGAHEAD, a cooling schedule typically starts at a peak and then drops. This would correspond to a single shake-up as the unit is at its most risk taking when it is at a peak in the cooling schedule. Here there is a single shake-up at the onset, and then the risk-taking tendencies of the commander slowly decrease over time such that the commander is completely risk-averse by the time the task limit is reached. Alternatively, the number of shakeups is set to 2, 3 or 4. This corresponds to the number of peaks in the cooling schedule. When there are multiple shake-ups there is always a shake-up at time 0, and the remaining shake-ups are evenly spaced across the duration of activity as defined by the task limit. So when a four shake-up strategy is employed the unit responding to tasks for 80,000 periods, faces shake-ups at task 0, 20,000, 40,000, and 60,000.

This virtual experiment is described in Table 1. Defining the domain values of the inputs yield 512 different experimental conditions. This is the portion of the parameter space that is to be explored.

Table 1: Summary of Input Parameters

Input Parameter	**Domain Values**
Task limit	20,000 and 80,000
Task complexity	binary and trinary
Task information	7 and 9
Agent ability	5 and 7
Environmental stress	Yes (stable) and No (periodic))
Unit Size	9, 12, 18, and 36
Shake-ups	1, 2, 3 and 4

4.3 *Stage 3: Set non-key inputs*

Non key inputs should be set to be random or should be set to match conditions known to be true of human groups.

Example: ORGAHEAD has been parametrically constrained in many other ways. The primary constraint is that only the relational aspect of the C^2 architecture is allowed to change; this means agents cannot be added or removed. Hence in the following virtual experiment, we ask how meta-learning occurs for organizations whose size is fixed. Within this and the aforementioned parameters, each unit's C^2 architecture was randomly generated for each run. Changes are allowed to occur every 500 tasks and performance is measured for the same duration as well. Memory capacities of the agents are set to 250 tasks, including that of the commander's look-ahead capability. In assessing whether a given change may benefit the organization, commander can look as far as 250 tasks ahead.

4.4 *Stage 4: Run simulations in the virtual experiment.*

Now you are ready to run the simulations. For each condition, each cell in the table describing your experiment, you should run multiple simulations. This is because there are stochastic elements. If you have a deterministic model you run each condition once. These simulations are your virtual experiment. In general the number of observations generated via a virtual experiment will be much larger than that generated via a human laboratory experiment, or in a gaming or live simulation situation.

Example: Using ORGAHEAD the space described by table 1 was explored by running a virtual experiment. For each condition shown in table 1, 40 units were simulated, per the Monte

Carlo technique. Output data was collected for performance, re-assignments and re-taskings. This virtual experiment resulted in 20480 data observations.

4.5 *Stage 5: Statistically analyze results.*

Computational models generate more data than human laboratory experiments. Nevertheless the results should still be statistically analyzed. Since there is so much data, it is possible to conduct multiple explorations.

Example: First the impact of the meta-adaptation strategies on influencing performance and sustained performance despite environmental and internal stressors was explored. The second exploration looked at the impact of the meta-adaptation strategies on the ultimate form of the resulting organizational or C^2 architecture. An indicator of this form is given by the difference in number of re-assignments and re-taskings. The findings should be treated as a series of predictions that can be explored in human laboratory experiments and with field data, or data from live simulations or games. In the following discussion, illustrative statistical analyses are done. Note that regression analysis and graphical depiction are both used as means of exploring the data.

For these analyses, results indicate that, in order of impact, the four factors which most affect sustained performance are: (1) the number of resources available to each agent, (2) the size of the unit, (3) the length of (amount of information in and resources associated with) the task, and (4) the number of shake-ups. These results are summarized in Table 2. Environmental stressors (Figure 1), not surprisingly, tends to reduce unit level performance (significant difference at the $p < 0.01$ level). However, the effect of more shake-ups is not strictly linear. We observe that performance slightly increases when the number of shake-ups increases from 1 to 2. Then, a significant drop occurs between 2 and 3 ($p < 0.05$). This degradation is somewhat restored when the number is once again increased to four. One possible explanation for this is that the effect of the shake-up strategy is sensitive to the particular kind and frequency of environmental variation. That is, we should see different effects if the periodicity with which the environment changed did not co-incide with the periodicity of the shake-ups.

Table 2. Standardized Regression for Performance.

Predictor	**Coefficient**	**p value**
intercept	0.000000	1.000
Task limit	0.031853	0.000
Task complexity	-0.024068	0.000
Environmental stressors	-0.014568	0.027
Unit size	0.170226	0.000
Agent ability	0.265205	0.000
Task information	0.091118	0.000
Shake-ups	-0.012299	0.063
R2 (adj) = 10.9%, df = 7, 20472, $p < 0.001$		

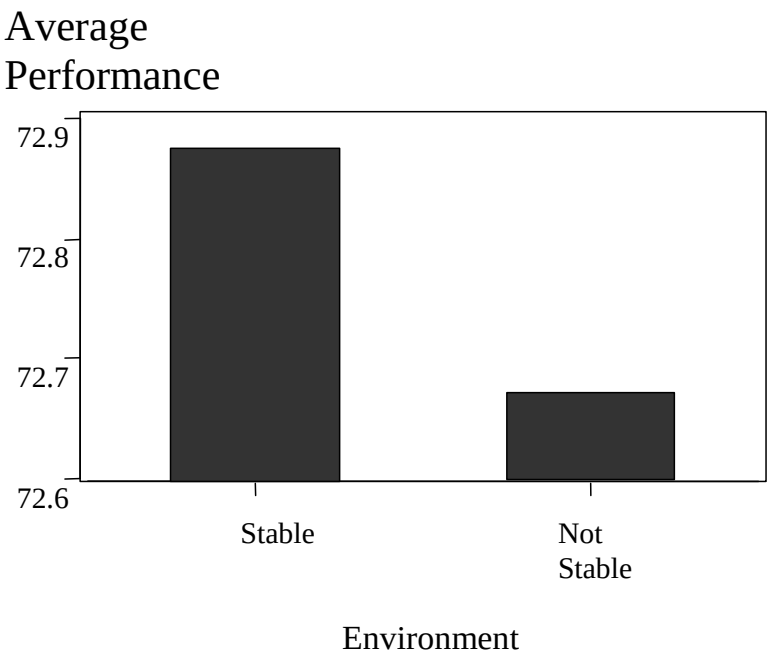

Figure 1. Impact of environmental stress on performance

In the changing environment the unit has adapted, or become synchronous, to the variation, while under the stable environmental condition, that goal is more difficult to achieve. Further, the number of personnel per level strongly determines the effect of the tuning process; we find that tuning yields more variation in performance for smaller units. Larger units are typically better able to withstand both internal and external stressors. Units that are too large (i.e. size 36) exhibit behavior of smaller organizations; their performance degrades when the environment changes.

Since there are only a finite number of changes that can be made, the type of change made is indicative of the way in which the unit adapts given a set of constraints. After controlling for task duration (obviously 80,000 tasks will yield far more changes), we look at the difference between the number re-assignments (people-to-people changes) and the number re-taskings (people-to-task changes). Both increasing organizational size and increasing the task complexity from 7 to 9 bits, reduces the number of re-assignments made and increases the number of re-taskings (Table 3). The first part of this finding is quite non-intuitive. If there are more people, then the probability of a re-assignment should increase. However, we find this number decreases implying that units are adapting by creating more direct linkages between personnel and task thus reducing the complexities brought on by inter-organizational communication. As a side result the amount of information and the number of resources available to any one individual increases.

Table 3: Standardized Regression for re-assignment minus re-tasking changes per task cycle.

Predictor	Coefficient	p value
intercept	0.000000	1.000
Task limit	-0.005913	0.396
Task complexity	0.009433	0.176

Environmental stressors	0.002727	0.695
Unit size	-0.084875	0.000
Agent ability	0.005552	0.425
Task information	-0.021738	0.002
Shake-ups	-0.004065	0.559
R2 (adj) = 0.8%, df = 7, 20472, p<0.001		

6 ***Stage 6: Generate Hypotheses***

Computer-based simulation, or computational analysis, now becomes a tool for generating hypotheses. Essentially, the main findings from the statistical analysis become hypotheses. Sensitivity analyses and studies of model robustness set limits on these hypotheses. Researchers can then test these hypotheses using data collected in the field, from archival sources, in human experiments. Researchers can determine whether these hypotheses also hold in other computer-based simulation models. This is a form of docking [Axtell et al. 1996].

To see how hypotheses are generated return to Table 2. In this table, here are a number of results relating both to what affects performance and to the relative impact of different factors. To generate a hypothesis, the statistical results are interpreted as a statement about how one or more of the input parameters relates to one or more of the outputs. Following are a set of hypotheses that follow from Table 2. These hypotheses are meant to illustrate the types of hypotheses possible, and not to enumerate all possible hypotheses.

H1: Units performing more limited tasks should exhibit higher performance than those performing less limited tasks.

H2: For a set of units, the higher the level of task complexity the lower the level of unit performance.

H3. As unit size increases so does unit performance.

H4: Hiring highly capable personnel should improve organizational performance more than increasing the amount of available task information.

H5. Personnel capability will lead to a 1.5 time greater improvement in performance than will increasing the size of the organization. Thus, organizations who retrain, rather than hire new personnel should do better.

Notice that these hypotheses range from simple predictions of correlations, to statements about the shape of a distribution, to statements of relative impact, to quantitative statements about impact. In essence, any type of hypotheses can be generated via computational analysis. Research done to test these hypotheses serves as validation for the model. What kinds of hypotheses can be generated depends on the type of model and the way in which the data from the virtual experiment is stored and analyzed. Quantitative hypotheses, like H5, are typically only made from semi-validated models, as part of an on-going research agenda to further validate a model.

5 Summary - Value of this computational approach to generating hypotheses

Computer-based simulation is a valuable technique for generating hypotheses. As the previous discussion illustrates, application of good experimental design procedures to collecting data from a computational model results in data that can be analyzed to generate a wide number of hypotheses

all of which are consistent with the underlying processes. Computational modeling allows the analyst to examine a larger number of parameters and to examine values or processes that may be impossible to examine in the human laboratory due to cost or ethical considerations. In this way, computational models are virtual laboratories and serve both a theory building and educational need [Bainbridge, 1987]. Computational models are ideally suited to the examination of dynamic systems and to suggesting the long term impacts of new technologies. Another advantage of computational analysis is that it enables analysis of groups far larger in size than can be analyzed in a field setting. As such, simulations are in essence tools for doing theory development. Computational process models are not, however, a panacea. There are limitations to their usefulness and there are conditions where they are more useful than others. A disadvantage is that such models cannot be used to conclusively demonstrate what people do in novel situations.

The areas where computational process models are most useful are:

1. The system is so complex that even a simple description involves a large number of input parameters.
2. There are important non-linearities in the processes.
3. The input parameters interact in multiple ways.
4. There are complex interactions involving three or more input parameters.
5. The analyst's interest is in the dynamics of the system.
6. The team or group being examined is composed of more than 3 personnel.
7. The team or group being examined is engaged in a knowledge intensive tasks.

Historically it was possible to test computational process models by doing a comprehensive analysis of the impact of all parameters. Current process models are sufficiently complex and veridical that a complete sensitivity analysis across all parameters cannot be done; rather, researchers often use response surface mapping techniques, experimental designs and statistical techniques to examine key aspects of the models. One of the key areas of research is how to validate and test these highly complex models.

One technique for validation is hypotheses validation [see also Carley, 1996b; Carley, Prietula and Lin, 1998]. Once the hypotheses have been generated, they can then be tested in various settings. One issue is what to do if the hypotheses is not validated. There are several reasons that this might occur. Most notably, the model may be wrong or the data may have been collected from a human setting using different measures or different conditions than in the model. Thus the first step is to check and make sure there is a match between the real and virtual world. If there is a match then the model is wrong and needs to be adjusted or discarded.

References

R. Axtell, R.Axelrod, J. M.Epstein, and M. D.Cohen. Aligning simulation models: A case study and results. Computational and Mathematical Organization Theory, 1(2) (1996), 123-142.

W.S. Bainbridge. Sociology Laboratory: Computer Simulations for Learning Sociology, Wadsworth Publishing Company, Belmont, CA, 1987.

H.H. Baligh, R.M.Burton, and B.Obel. "Devising expert systems in organization theory: The organizational consultant," in Organization, Management, and Expert Systems, M. Masuch, (Editor), Walter De Gruyter, Berlin, 1990, pp. 35-57.

R.M. Burton and B. Obel. The Validity of Computational Models in Organization Science: From Model Realism to Purpose of the Model. Computational and Mathematical Organization Theory. 1(1) (1995), 57-72.

K.M. Carley. Adaptive Organizations: A Comparison of Strategies for Achieving Optimal Performance, Proceedings of the 1996 International Symposium on Command and Control Research and Technology. June. Monterray, CA, 1996a.

K.M. Carley. A Comparison of Artificial and Human Organizations, Journal of Economic Behavior and Organization. 31 (1996b), 175-191.

K.M. Carley. Organizational Adaptation, Annals of Operations Research, 75 (1998) 25-47.

K.M. Carley. "On the Evolution of Social and Organizational Networks," in special issue of Research in the Sociology of Organizations on Networks In and Around Organizations, vol. 16, S. B. Andrews and D. Knoke (Editors), JAI Press, Inc. Stamford, CT, 1999 pp. 3-30.

K.M. Carley and C. Butts. An Algorithmic Approach to the Comparison of Partially Labeled Graphs, Proceedings of the 1997 International Symposium on Command and Control Research and Technology. June, 1997, Washington, DC.

K.M. Carley and L. Gasser. "Computational Organization Theory," in Distributed Artificial Intelligence, G. Weiss (Editor), MIT Press, Cambridge, MA, 1999, ch. 7.

K.M. Carley and A. Newell. The Nature of the Social Agent, Journal of Mathematical Sociology, 19(4) (1994) 221-262.

K.M. Carley, M.J. Prietula, and Z. Lin. Design versus Cognition: The Interaction of Agent Cognition and Organizational Design on Organizational Performance, Journal of Artificial Societies and Social Simulation, 1(3) (1998) 1-19. 30 June 1998 at <http://www.soc.surrey.ac.uk/JASSS/, 1: paper 4.

K.M. Carley and J.S. Lee. "Dynamic Organizations: Organizational Adaptation in a Changing Environment," in Advances in Strategic Management, vol 15, Disciplinary Roots of Strategic

Management Research, Joel Baum (Editor), JAI Press, , Inc. Stamford, CT, 1998. Ch. 15, pp. 267-295.

K.M. Carley and Z. Lin. A Theoretical Study of Organizational Performance under Information Distortion, Management Science, 43(7) (1997) 976-997.

K.M. Carley and D. M. Svoboda. Modeling Organizational Adaptation as a Simulated Annealing Proces, Sociological Methods and Research, 25(1) (1996) 138-168.

J. Epstein and R. Axtell. Growing Artificial Societies, MIT Press, Boston, MA, 1997.

Y. Jin and R.Levitt. The Virtual Design Team: A computational model of project organizations. Computational and Mathematical Organization Theory, 2(3) (1996), 171-196.

D. Krackhardt and K.M. Carley. A PCANS Model of Structure in Organization in Proceedings of the 1998 International Symposium on Command and Control Research and Technology. June. Monterray, CA, 1998.

Nowak and B. Latane. Simulating the energence of social order from individual behaviour, in Simulating Societies: The computer simulation of social phenomena, Gilbert, N. and J. Doran (Editors), UCL Press, London, 1994, pp. 63-84

.

Technical Biography

KATHLEEN CARLEY

Kathleen Carley received her Ph.D. from Harvard University in Sociology in 1984. Prior to that she received two S.B. degrees from Massachusetts Institute of Technology in 1978. She joined Carnegie Mellon University in 1984 as an assistant professor, and in 1998 became a professor of Sociology & Organizations. She has a joint appointment in the Social and Decision Sciences, H.J. Heinz III School of Public Policy and Management, Engineering and Public Policy, and the Institute for Complex Engineered Systems. She is the director of the center for Computational Analysis of Social and Organizational Systems (CASOS) at CMU. Professional memberships include Informs, AAAS, Sigma-XI, ASA, and INSNA. She is currently the President Elect for ASA Mathematical Sociology Section. She has published over 70 articles and 2 books employing computational analysis, is a co-editor of the journal Computational and Mathematical Organization Theory, and is a co-organizer of the annual workshop in Computational and Mathematical Organization Theory.

[1] An emulation model is a computer-based simulation model in which the parameters can be set to emulate exact features of the system being modeled. For example, in a model of a team, in an emulation model the task precedence relations, the number of humans on the team, differences in training, etc. that actually occur in the real hum team would be entered as input to the computer-based model. An example of an emulative model is the Virtual Design Team, VDT [Jin and Levitt, 1996].

[2] An organization's architecture or form can be represented by a series of inter-locked matrices relating personnel to resources/knowledge and tasks [Krackhardt and Carley, 1998]. Given two such architectures, even if they differ in size, canonical labeling can be applied to create a role independent label for each node [Carley and Butts, 1997]. Then the nearness of the two organizations can be calculated as the hamming distance.