Canonical Task Environments for Social Simulation

Scott Moss
Centre for Policy Modelling
Manchester Metropolitan University
s.moss@mmu.ac.uk
http://www.cpm.mmu.ac.uk/~scott

1 Introduction

The purpose of this paper is to propose and describe an alternative to an overarching theory for social simulation research. There are a number of reasons why some general framework will be useful.

One is that the social simulaton community seems to produce a bespoke model for every situation or issue modelled. The evidence is not hard to find. The only attempt explicitly to relate models to one another was reported by Axtell *et al*. [1]. The results reported in that paper were achieved by means of a collaboration between Axtell and Epstein at Brookings and Axelrod and Cohen at Michigan to demonstrate that their respective models yield the same results when applied to the same social situations.

One reason for *aligning* models in this way is that, informally, we have more confidence in results obtained from a wide range of model specifications than we would if different models gave contradictory results. To the extent that we want to search for results that are robust with respect to the social situation or with respect to the particular representaton of agent cognition in a given social situation, then it is clear that the accumulation of model alignments in this way is essential in all circumstances where analytical results are not available.

A second reason is to be able to determine when or whether agent representations of modelling techniques used in the analysis of one issue can be used in the analysis of others. In extending domains of application of models, is it necessary to change representations of cognition channels of interaction among agents and, if so, how?

In summary, an important guide to the direction of social simulation research and application would be some means of situating models relative to one another and relative to domains of application.

While the procedures and results reported in [1] represent an important step in this direction, the effort involved seems to have been considerable, involving large amounts of programming and personal visits between the two institutions. The result was a useful and impressive demonstration of the possibilities of "docking" or "aligning" models developed in different software environments for different purposes. Moreover, those authors were interested in determining one relationship among models: subsumption or what econometricians call nesting. This means that one model is a special case of the other.

The development of what are called below provides a basis for model alignment and situating which can be more general than subsumption and which is a much more economical approach than that used by Axtell *et al.*

2 Canonical task environments

In mathematics, canonical matrices are effectively matrices of a standard form and there are transformations which can be performed on other matrices to show that they can be made into canonical matrices. All matrices which, by means of allowable operations, can be transformed into a canonical matrix have the properties of the canonical matrix.

So a canonical task environment in social simulation will naturally be defined as an abstract form with known properties that can be used to represent environments and the effects of cognitive behaviour in such a way that models with particular domains of application can be mapped unambiguously into models set in canonical task environments.

The virtues of canonical task environments will be demonstrated by using it to capture three computational models of organizational behaviour. These are the Carley-Svoboda [3] model in which workers in an organization have the task collectively to recognize the modal digit in a digit string; the Moss [8] model of the resolution of critical incidents by an environmentally sensitive organization and the virtual design team (VDT) model [7] in which agents have to cooperate in order to achieve a design task. The Carley-Svoboda model incorporates a recognition task; the Moss model an independent action task and the VDT model a cooperative acton task. The Moss and VDT models are distinguished in this regard by latter's incorporation of a critical path model to describe the necessary work programme for the design task while, in the Moss model, any sequence is possible and effective sequences emerge as a result of agent cognition and behaviour.

All of these models are intended to capture elements of agent cognition in organizations. The Carley-Svoboda model relates organizational structure to effective agent cognition; the Moss model relates communication among agents to effective organizational performance and the VDT model relates both organizational structure and communication to performance. So, although the models are implemented in different ways and represent cognition differently, they are clearly in the same domain. The verbal description of their areas of application suggests that the Carley-Svoboda task environment is nested in the task environment of the Moss model which in turn is nested in the task environment of the VDT model.

One purpose of this paper is to state that set of nesting (or subsumption) relationships analytically. This is done by showing that both the Moss and VDT task environments map directly into the Carley-Svoboda specification of the task environment. This, for reasons to be considered in the section following, is the starting point for the canonical task environment developed below. In order to maintain as much comparability as possible among the implementations of the three models, a single base representation of cognition will be employed, although some differences in implementation are required for each model. These differences and the reasons they are required are themselves informative in relating these models to one another. Finally, although a simple matrix-based set of relations among the aspects of the environment and the relationships among actions and aspects of the environment is implemented here, generalization of the techniques to arbitrarily complex relationships among aspects of environmental states and agents' actions will be described though not implemented here.

3 Developing a canonical task environment

The customary form of reporting models and results is as if a problem were identified and the model used to analyse that problem appeared like the 7th Cavalry riding to the rescue. In this case, an account of the development of the suggested canonical form of task environment robot will itself help to demonstrate its canonicity and to suggest an approach to the development of canonical task environments in general.

The inspiration for the model described here is a set of models by Kathleen Carley and her colleagues which represent the environment as a digit string and model agents with the objective of recognising certain characteristics of that string such as whether it contains more 1s or 0s. The key paper here is by Carley and Svoboda [3]. This model has much the same content as the Radar-Soar model [15] in which radar station analysts were observing airplane characteristics in order to determine whether they were friendly, neutral or hostile. The airplanes had nine observable characteristics each of which could take values of low, medium or high. Agents were either analysts or managers. In both cases, cognition was represented in Soar, the computer architecture manifestation of Alan Newell's [12]unified theory of cognition. Several alternative formulae were applied to relate the characteristic values to the "actual" status of the airplane as friendly, etc. There is no analytical difference between representing an airplane as a list of tokens from the set {low, medium, high} on the one hand or a list of digits from the set {0, 1, 2} on the other. In fact, converting the tokens into numbers was one of the steps involved in calculating the status of the aircraft.

This gives us an initial representation of the environment for a canonical model: the digit string. In all of these models, organizational structure is represented by specifying agents that observe subsets of charcteristics or, equivalently, positions in the digit string and the agents to which the observing agents report and the agents to which they report, and so on. Decision-making within that structure is imposed by the modeller or at random or, in the latest version, by a special agent (the CEO) who periodically modifies the organizational structure as a cognitive act. In all of these cases, the function of the organization is correctly to recognize some characteristic of the digit string. In the radar model, the recognition task is in practice to recognize in which of three ranges lay the value of a function of the elements of the digit string and in the Carley-Svoboda model to recognize whether a bit string contained more 1s or 0s.

A canonical task environment that is restricted to the representation of recognition tasks is clearly too limited to provide a focus for the issues of concern to the social simulation community as represented by, say, recent conference volumes in the field such as [5] and [6]. To extend the model to allow for agents to act is therefore a natural step.

In order to support social simulation modelling of real institutions and systems, it is essential to demonstrate that a canonical task environment supports application models of such institutions and systems in its domain. For this reason, the elaboration of the radar and digit string models to include action by agents entailed a mapping from a model of the organizational structure and systems involved in the management of critical incidents in an environmentally sensitive industry—the UK water and treatment industry. The model was based on interview and documentary data provided by North West Water PLC [8].

The relevant characteristics of the model are the representation of operating sites as non-cognitive agents that communicated by telemetry with the central systems. The telemetry data included alarm states for intruders, fires, floods, leaks of various kinds, pump failures, and so on. There was also information provided by the public concerning publicly observable events such as mains collapses or dead fish in the rivers. In each case the event could be occurring or not. So far, this application is not essentially different from the radar problem. The occurrence of an event is represented by a 1 and the non-occurrence by a 0. Ordering the 1s and 0s by operating site and event type (intruder, fire, mains collapse, chlorine leak, etc.) yields a digit-string representation of the environment. In addition, each of these critical events has associated with it a corrective action such as remove intruder, extinguish fire, repair leak. The simplest extension to the Carley-Svoboda model that will capture the possibilities for action to resolve critical incidents is a bit string s implemented as a vector and another bit string a, also a vector, of the same length as s and in which a 1 represents taking the action such as remove intruder at a particular operating site and a 0 indicates that the action is not being taken at that site. The effect of a 1 in a given position in the action string a changes the value of the corresponding digit of s from a 1 to a 0, unless it is already 0.

This gives us a straightforward difference equation for the state string:

(1) $\Delta s = -Ia$

The North West Water engineers allow for probabilities that action taken to resolve one event might create another so that, as a fictional example, in repairing a pump it is possible that a fire will be started. This implies that the coefficient matrix of equation (1) might have off-diagonal elements. Consequently, we require a more general form of the coefficient matrix: an action-state-change matrix (ASCM), to be denoted by A where $[a_{ij}]$ is the effect of the unit action at the ith digit of the action string on the jth digit of the state string. The notion of a probability or observed frequency distribution of such events is captured in the usual way by a probability matrix P where $[p_{ij}]$ is the probability that a change in the *i*th digit of the action string will have the specified effect on the *j*th digit of the state string

A further effect is that of "snowballing". One event might trigger another without any actions having been taken. The "snowballing" or state-state-change matrix (SSCM) will be denoted by *S*. There is no reason not to allow for each link in the sequence of state-state interaction to be subject to an observed frequency distribution and if that is the case we define an appropriate probability matrix.

While the SSCM will naturally be a square matrix of size equal to the length of the state digit string, we would not expect in general that there will be an action to affect every aspect of the environment or, in the present context, as many rows as columns of the ASCM. So if there are *a* actions and *s* defined aspects of the state of the environment, the ASCM will by an $a \times s$ matrix. So, ignoring snowballing, instead of equation (1) the change in state digits due to actions will be

(2) $\Delta s_t = A a_{t-1}$

In keeping with the evidence from North West Water, we assume that a change in a state digit at one period will affect the values of a some state digits at the next period so that, ignoring for the moment constraints on the values of the digits,

(3) $\Delta s_t = A a_{t-1} + S \Delta s_{t-1}$

Substituting recursively into the second term on the right,

(4) $\Delta s_t = A \sum_{\tau=0}^{T} S^{\tau} a_{t-1} + S^{T+1} \Delta s_{t-(T+1)}$

At period 0, the change from the "previous" period is 0 and so at period *t*, the change in the state string is

(5) $\Delta s_t = A \sum_{\tau=0}^{t} S^{\lambda} a_{t=\tau}$

Planning issues become important when the value of coefficients in the *A* matrix are larger in magnitude than the differences between current and target values of the state string digits they affect. Reaching the target values can only be done indirectly by changing related state string digits or by an indirect route of increasing the discrepancy from the target values in order to converge in some other way. This kind of planning is a natural application for representations of cognition both as modelling and as formal logics of any of the intelligent planning algorithms and to investigate how the efficiencies of these various approaches relates to organizational structures.

4 Action in the canonical task environment

In this section, the environments from two further models are mapped into the canonical task environment to demonstrate how to allow for agents to take actions to influence the environment. Moss' North West Water model [8] is concerned with actions that can be taken independently of every other. The VDT model captures certain features of critical path models including the requirement that some results require several actions to be completed in parallel or in a certain order. The Moss model gives to individuals independent control tasks while the VDT model gives to individuals some co-operative control tasks. In this section, we demonstrate that they are both in the domain of the same augmentation of the recognition task environment

4.1 Recognition task environment.

The Carley models as noted do not represent action by agents — only observation, reporting and formulating hypotheses about the implications of the observations. In terms of the canonical model, the action string contains all 0s and equation (4) becomes

(6) $\Delta s_t = 0$

In this model, nothing would ever change without some exogenous perturbation of the digit string. In order to represent changes in organizational forms and to analyse the effectiveness of different assumed structures in the recognition task, states in this model are changed by random mutations in the digits of the state string so that, in the recognition task model, the state change equation is

(7) $\Delta s_t = S\Delta s_{t-1} + m_t$

This yields the usual sort of geometric progression so that

(8) $\Delta s_t = \sum_{\tau=0}^{t} S^{\tau} \cdot m_{t-\tau} + S^{t} \cdot \Delta s_0$

Since the value of $\Delta s_0 = 0$ by construction, the change in the state string at any time t is simply

(9) $\Delta s_t = \sum_{\tau=0}^{t} S^{\tau} \cdot m_{t-\tau}$

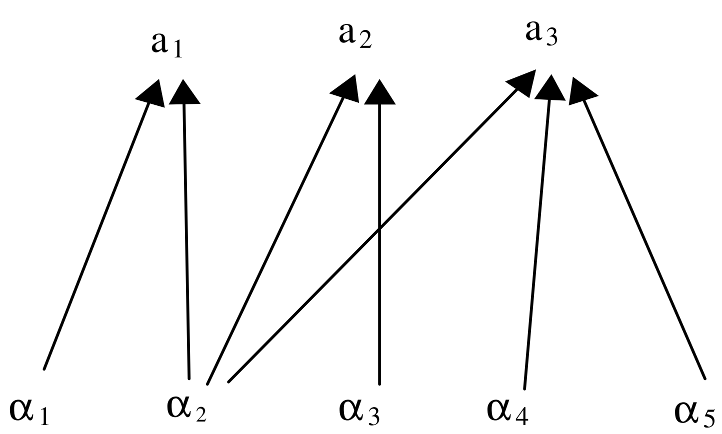
In all of the task environments discussed in this paper, there is a user-specified probability that any element of m_t will be non-zero and, if non-zero, an equal chance of being positive or negative. Changes in the state digit string are bounded so that the value of every element in the state string is a non-negative, single-digit number. The number base of the state string is therefore the number of categories in which each observed event could be placed. An increment of (say) 1 from the highest single-digit number is always 0 and a decrement of 1 from 0 is the highest single-digit number.

4.2 Independent control task model

This is the model already reported. In the simulation runs reported in the next section, random mutation of the state string digits was implemented in order to generate some noise in the system. When the agents were successful in eliminating discrepancies between observed and target digits at the positions they observed, the noise was necessary to create new discrepancies and, so, use the cognitive capacities of the agents and their organizational structures. This is also consistent with the view of the critical-incident managers of North West Water that particular types of events have known probabilities of occurrence. The state change equation for the independent control task model is

$$(10)\quad \Delta s_t = \sum_{\tau=0}^{T} S^{\tau}(Aa_{t-\tau} + m_t)$$

Figure 1: Digraph representing relationship between α and a strings



4.3 *Co-operative task control model*

A co-operative control task model effectively requires a network relationship among actions and targets. In order to see what is involved here, and how it is incorporated into the canonical model, we interpret the action string of equations (5) and (10) as intermediate outcomes of lower-level acts denoted by the string a. Suppose that there are three intermediate outcomes represented by the digit string **a** = [a_1, a_2, a_3] and three direct actions available to agents represented by the values of the digit string α = [α_1, α_2, α_3, α_4, α_5]. The relationship between the values of the elements of α and the elements of a are given by the digraph in Figure 1.

The corresponding edge matrix is

$$(11)\, E = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

We define the operator $\oplus$ so that

$$(12)\, E_i \oplus \alpha = \min(\alpha_j \mid E_{ij} = 1)$$

This is a perfectly standard path algebra so that we could allow for as many links among the α nodes leading to the *a* nodes as we wished. Two steps in the traverse from a leaf node to an *a* node would be determined by $E \oplus E$ and three steps by $E \oplus E \oplus E$, and so on. Each of these steps would be assumed to take one appropriate time period.

Allowing only for single links between the α nodes and the *a* nodes,

$$(13)\, a_t = E \oplus \alpha_i$$

The state change equation is then

$$(14)\, \Delta s_t = \sum_{\tau=0}^{T} S^{\tau} (A \cdot E \oplus \alpha_{\tau} + m_t)$$

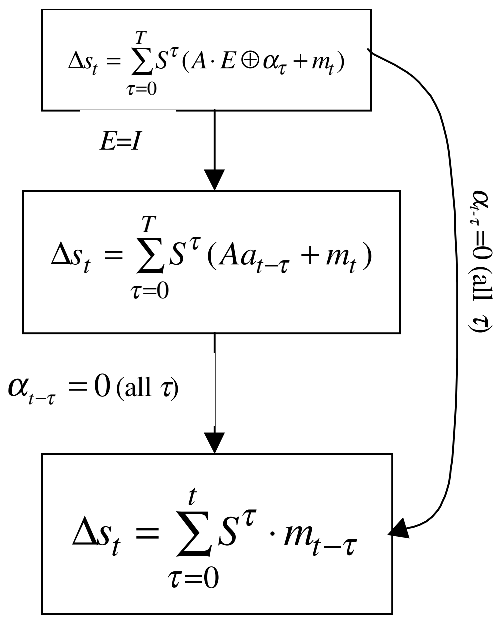
4.4 *The canonical task environments: a summary*

The relationships among the canonical task environments defined so far are easily seen in Figure 2.

The cooperative task environment is reduced to the independent task environment by conflating the first level of actions taken by individuals with the "levers" on the environment. Formally, this amounts to allowing only one level of actions which is equivlent to setting the edge matrix equal to the identity matrix. To reduce the task environment further to the recognition task environment, we simply allow there to be no non-zero values of the actions. This will reduce either of the more general task environments to the recognition task environment.

The relationship among models whereby one is reduced to another by setting individual parameters to specific values is known in econometrics as nesting and is, I believe, what Axtell *et al.* had in mind by subsumption. Consequently, the argument of this section indicates that the co-operative action environment subsumes the independent action task environment which in turn subsumes the recognition task environment.

Figure 2: Relationships among the three canonical task environments



5 The scope of simulation results

While it is important to investigate relationships such as subsumption among models and the canonical task environments will certainly facilitate such investigations, it is also important to determine the scope of the results obtained from individual models. By scope of the results, I mean the range of canonical environments over which a set of results will be replicable. For example, Carley and Svoboda obtained a number of results about organizational structure in the recognition task environment. Those results are themselves clearly more robust if they are also found in the independent and cooperative action environments. Moreover, and perhaps more importantly, if we find that results obtained in the recognition task environment typically can be replicated in the more complicated task environments, there will be obvious computational economies to be had in simulation experiments.

In order to identify those results that are dependent on the task environment, it is obviously important to implement the other features of models in common ways. Clearly there will have to be some differences. Agents in the recognition task environment will not require the capacity to act and agents in the independent action task environment will not require the capacity to co-operate while action and co-operation both are required in the co-operative action task environment. In order to minimize the differences in results due to implementations of agents, the agent specifications will have to be nested in a manner corresponding to the nesting of the task environments.

Accordingly, the three canonical forms of the task environment are implemented in SDML, using its object oriented features to ensure that these are nested in the manner identified in section 4. The purpose is to investigate the scope of the results obtained in the original models that inspired the canonical representations of the task environment in the first place.

SDML supports three hierarchies with object oriented features. Two of these hierarchies support multiple inheritance and one supports simple inheritance. The top-level hierarchy is the *module* hierarchy. Within each module is a hierarchy of *types*. The type hierarchy entails the definition of *containers* and a hierarchie of agents containing other agents. The module and type hierarchies support multiple inheritance and the container hierarchy supports simple inheritance.

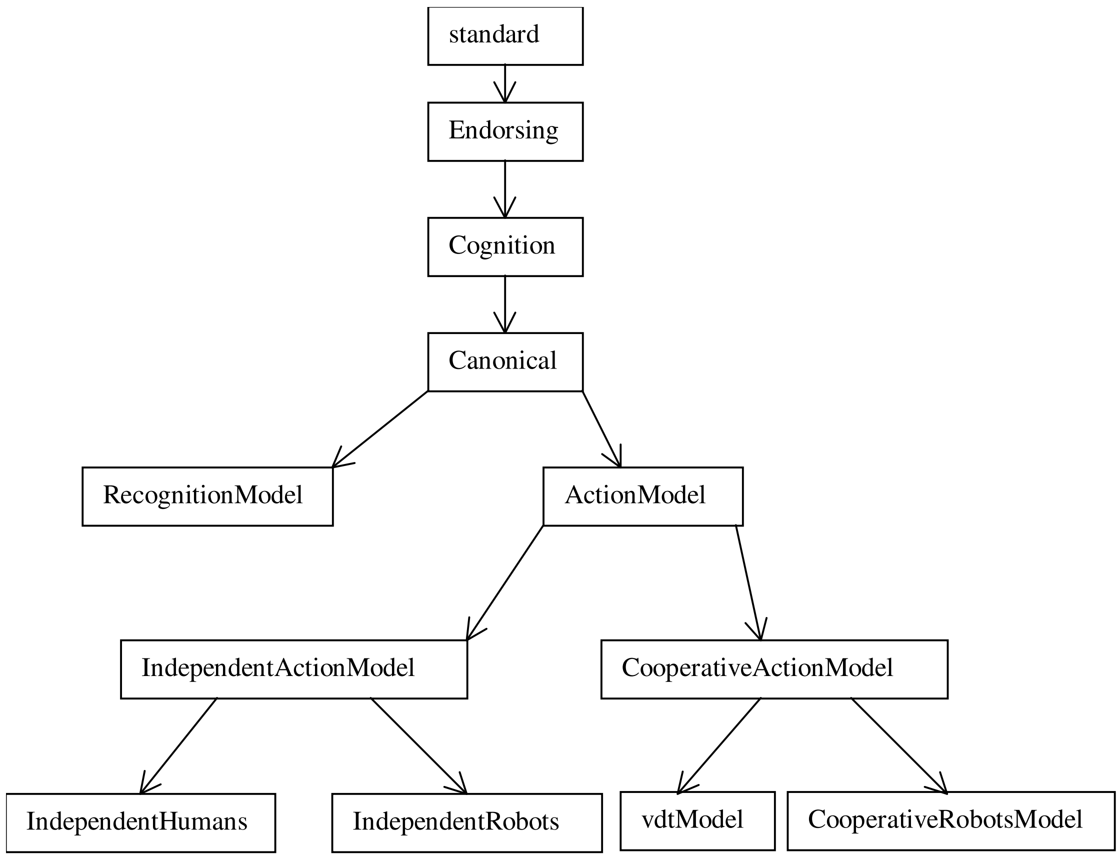
Models in SDML are associated with modules. The modules associated with models for the various task environments will require either to contain or inherit both the functionality required for the task environments and the representations of agents with cognitive capacities relevant to their respective task environments.

5.1 The module hierarchy

The module hierarchy for the development of the task environments is reproduced in Figure 3. The top-level module, *standard*, contains the essential functionality of SDML and cannot be modified by the user. All of the other modules must be submodules of *standard* and, so, inherit the essential SDML functionality.

The two modules, *Endorsing* and *Cognition* implement a representation of decision-making and mental modelling that has been used in a large number of models including those reported by Moss (1998), Moss and Sent (1998), Moss and Dautenhahn (1998), and Moss, Gaylard, Wallis and Edmonds (1997). The *Endorsing* module contains the implementation of a development from Paul Cohen's (1985) endorsements scheme. Agents endorse information, other agents, mental models, rules of behaviour, information sources or other objects of the model. The endorsements are mnemonic tokens which have positive values if good or negative values if bad. The magnitude of the value associated with each endorsement token indicates its class of importance. This is used to calculate the overall endorsement value of one object in order to compare it with the endorsement values of other objects.

Figure 3: The task environment module hierarchy



Every endorseable type of object (rules, mental models, *etc*.) has associated with it an *endorsement scheme* which defines the endorsement tokens and values as well as a number base for calculating the values of all endorsements of an object. If the number base is b, the endorsement valueof an object will be

(15) $E = \sum_{e_i \geq 0} b^{e_i} - \sum_{e_i < 0} b^{|e_i|}$

where e_i is the value of the ith endorsement token. Clearly, the value of an endorsement token in class n is b times as important as a token in class (n-1). The finer the grain of endorsement values, the closer b is to 1.

An example of an endorsement scheme is reproduced in Figure 4 from the SDML rule setting up the endorsement scheme for the workers in organizations in the action models.

Figure 4: SDML rule setting up plan endorsement scheme

```
and
  = ?grain 1.3\
  =  ?endorsementValues
        [[planSuccessful 3]
         [planOnTrack 2]
         [planOffTrack -1]
         [planWrongInDirection -2]]
```

```
permanent
  (endorsementSchemeDefinition
        planEndorsementScheme
        ?endorsementValues
        ?grain)
```

The grain of the endorsement scheme (b in equation (15)) is 1.3. The variable ?endorsmentValues is set equal to the list of tokens and their respective values. The object planEndorsementScheme is then asserted to the permanent database so that each worker in the model can access it at any time during a simulation run.

Because this rule is defined for the cognitive element of agents of type Worker in the ActionModel module, all agents of that type in every task environment providing for agent actions will have the same endorsement scheme for plans of action.

The Cognition module implements a representation of agent cognition taken from Soar and ACT-R at the computational end of cognitive science. This is based on a problem space architecture with a set of tasks being associated with each problem space. The problem spaces are *alarm*, *noPlanExists* and *communicate*. In other models, the problem space architecture has been considerable more elaborate. A cut-down version is used in the models reported here in order to maintain the issues assoicated with canonical task environments in the sharpest possible relief.

The problem space alarm arises when a worker notices that the value of the state string at a position observed by that agent differs from some target value. The objective of the agent is then to change the value of the state string digit to its target value. The first question is whether the agent has a plan defined that will be applicable in the current circumstances and which has as a goal the change in the value of the state string at the specified position. If not, then the agent enters the problem space *noPlanExists* and then has a number of rules to guide the creation of a plan. If there is an appropriate plan, or once one is defined, the agent is no longer in the *noPlanExists* problem space and the problem space *communicate* becomes apposite. In that problem space, the worker communicates with a superior in the organization to inform that superior of the action taken or that no action could be taken but that the observed digit of the state string is not at its target value. The possible components of the plans will differ with the model but the problem space architecture and the plan endorsement scheme will not. If there are several possible plans, the chosen plan will be one with the highest endorsement value.

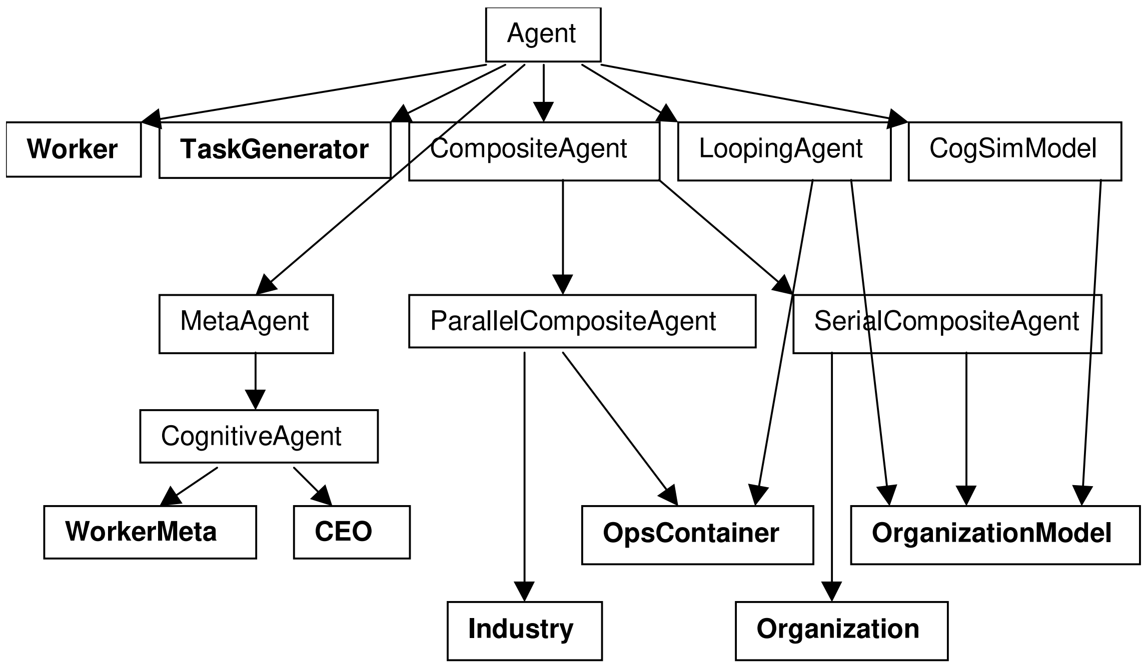
Inspection of the module hierarchy will confirm that the extent of such common properties of the various models is substantial. The problem space architecture and the endorsements mechanism have been defined in this way because they are compliant with Soar and ACT-R cognition as used in social simulations by Ye and Carley (1995) and Moss (1998) and because they allow for the specification of the problem spaces and associated tasks as well as the endorsements and their values on the basis of data supplied by domain experts. The expert input is not relevant in this case but it is by no means inconsistent with the canonical task environments.

5.2 The type hierarchy

Type hierarchies are augmented by modules but all types and their subtypes defined higher in the module hierarchy are inherited. In the present case, there is one type hierarchy defined in the Canonical module (see Figure 3). In the type hierarchy are two types which can be changed by users and for which subtypes can be defined. These are the types *Object* and *Agent*. In fact, Agent is a subtype of Object. Its added functionality is in having rulebases and databases and the ability to access them. Similar type hierarchies to that implemented for the models set in canonical task environments have been reported several times previously (*e.g.* in Moss (1998)).

The segment of the type hierarchy starting with type Agent and including its subtypes that are relevant to the substance of the models set in canonical task environments is reproduced in Figure 5. The types in boldface are defined in the module Canonical. The types CognitiveAgent and CogSimModel are defined in the module Cognition. The other types are primitive to SDML and are inherited from the module standard.

Figure 5: Segment of type hierarchy from the Canonical module



In the lower left corner of Figure 5 are the type MetaAgent and its subtypes. Instances of type MetaAgent are unique agents in that they have not only their own databases and rulebases but, in addition, they can treat the rulebases of certain other agents as databases. That is, instances of type MetaAgent and its subtypes can read the rules on certain rulebases and they can also write rules on those rulebases. In all of the models reported here, each agent of type Worker *contains* an instance of type WorkerMeta. The instance of WorkerMeta devises mental models and, on the basis of those mental models, devises plans of action. These plans of action comprise sequences of actions and the conditions in which those actions will be taken. The actions are actually taken by the containing instance of Worker because the conditions and actions are written as rules to the rulebase of the Worker instance.

In general, any agent can contain a meta agent and any meta agent can read from and write to the rulebase of its containing agent, or *container*. Instances of the type CEO are the mea agents of instances of Organization. Each organization contains one CEO instance which, following Carley and Svoboda (1996), reorganizes the structure of the organziation periodically in order to improve its effectiveness.

Instances of LoopingAgent and its subtypes function over nested iterations corresponding to time levels. The instances of ParallelCompositeAgent and its subtypes contain subagents that fire their rules in parallel at each time period. The instances of SerialCompositeAgent and its subtypes contain agents that fire their rules in a specified order.

The combination of parallel and serial behaviour with time levels is most easily seen in relation to the container hierarchy.

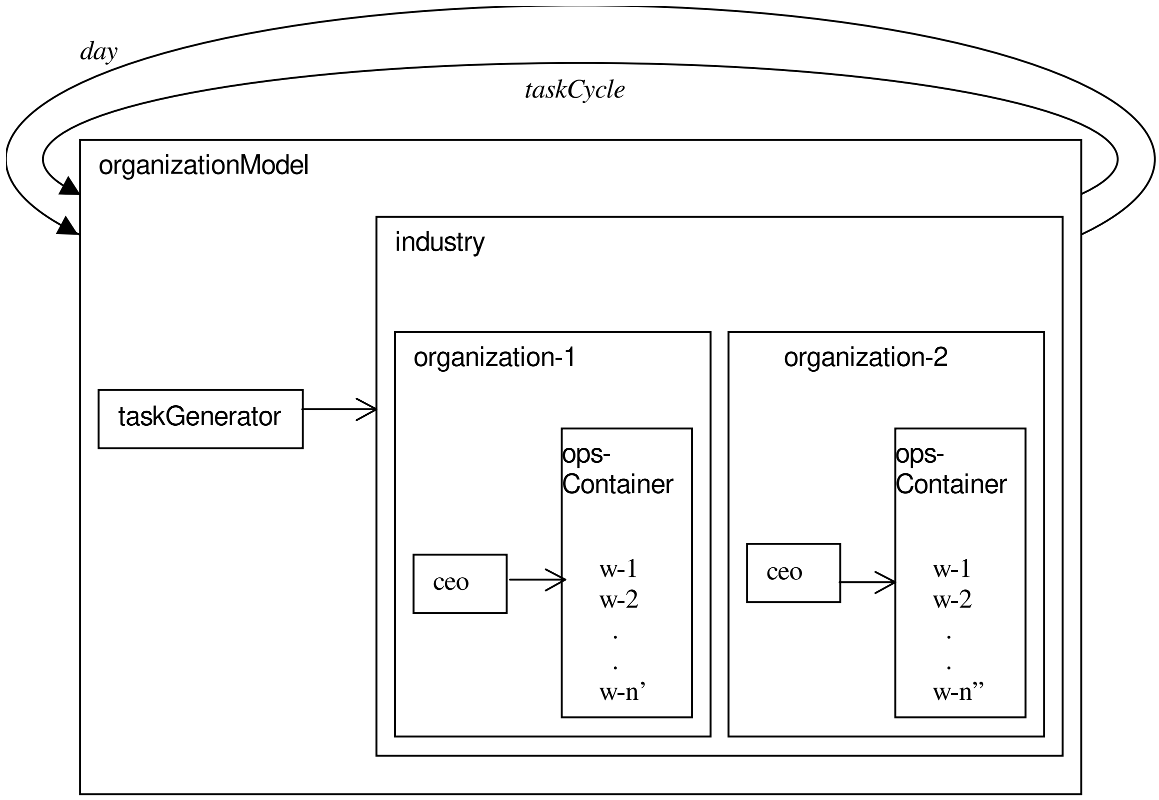
5.3 The container hierarchy

The outermost container in SDML is always *universalAgent* which is the only instance of type *Universe*. Usually it contains only a single subagent which is an instance of a user defined type of model. The container hierarchy defined by the types in module Canonical is reproduced in Figure 6.

The model has two active subagents: taskGenerator and industry. Industry can contain an arbitrary number of organizations. In all of the models reported here, there was only one industry but, in Figure 6, two are shown to indicate better the capacity of the setup. Each organization contains a ceo and an operations container called opsContainer. The rulebases of ceo are active before the rules of opsContainer and its subagents. The rulebases of the workers contained by opsContainer become active effectively in parallel. That is, no worker can "see" the actions or consequences of any actions of any other worker in the current time period.

At the start of each task cycle, taskGenerator collects all of the actions taken in the previous task cycle and calculates the consequential changes in the state string, incorporating any mutations which it also determines. After the state has been updated, the industry becomes active and the organization rulebases are activated in parallel. So the model cycles in this way over *taskcycles*. There are three *taskCycles* per *day*. The only effect of the distinction is that, at the start of each *day*, the ceo of each organization considers the need for reorganization on the basis of conflicting actions by workers in the organization observed during the previous *day*. The ceo has no other function and it has no active rulebases except at the start of the *day*.

Figure 6: Container hierarchy defined in module Canonical



In order to allow for communication among agents in the determination of actions, opsContainer in each organization cycles over a time level called *decisionCycle*. This enables agents to decide to refer some decision up to a superior or to issue an instruction to a subordinate so that the communication is received during the same task cycle. The need for this additional time level follows from the parallel nature of the actions of each worker in the organization. Consequently, one worker will “say” something to another worker (by asserting a clause to the receiving worker’s database) but the recipient of the message will not be able to observe it until the following time period – in this case the following decision cycle.

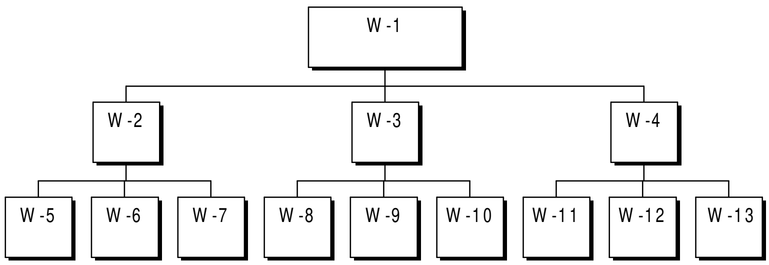
Before each worker activates its rulebases, its meta agent (if any) is activated. In the representation of cognition used here, each meta agent cycles over time level *elaborationCycle* during which it moves among its problem spaces to get as far as possible in deciding on the approapriate rules of action for its containing worker. This is explained more fully by Moss *et al.* (1997).

6 Experimental design

The purpose of the experiments reported below is to use the framework of the canonical task environments to assess the extent to which results obtained in (say) the recognition task environment extend to more complicated task environments.

The object-oriented features of SDML are used to ensure that there is a single implementation of as many as possible of the common features among task environments and models. Some features, such as organizational structure are not hard-wired. Indeed, the purpose of the experiments reported below is to identify any implications for organizational structure that extend over different task environments. So we start with a common organizational structure for all models as reproduced in Figure 7 and allow the CEO of the organization to alter the structure at the start of each day. This agent is allowed only to eliminate conflicting actions by workers and managers in the organization.

Figure 7: Initial organizational structure in all models



In the remainder of this section, the experiments with each of the canonical task environments are described and motivated.

6.1 Recognition task environment

The general innovative aspect of the Carley-Svoboda paper was that it combined the modelling of structural organizational change with learning by adaptive agents within the organization. Their findings focused on comparisons of organizational performance under individual learning with no change in organizational structure, structural change with no individual learning and dual-mode learning in which both structure can change as the CEO learns and individuals within the organization learn and, in this case, new individuals can be hired. Their conclusions in each case were based on simulations of 1,000 initial organizational forms over 20,000 task cycles each. Evidently, they simulated 60,000,000 recognition tasks.

The purpose of this section is not to conduct a replication experiment. It is, instead, to use the canonical task environments to determine how the results from one experiment fare under increasingly complex task environments. The recognition task environment provides our baseline. Experiments were run in which agents are adaptive and develop mental models of relationships between their observations of assigned state string digits and their predictions of the modal numeral in the string. The only possible representations of an increase in the complexity of the environment is an increase in the number of non-zero elements of the state-state change matrix (SSCM), an increase the number base of the state string or an increase in the length of the state string. Demonstration of the experimental value of canonical task environments will sufficiently entail increases in the interaction among digits in the state string while using only bit strings of different lengths. Via the endorsements mechanism, managers come increasingly to value subordinates who correctly guess the modal numeral and to ignore those who persistently guess the incorrect modal numeral. The observer-workers generate and test relationships between their observations and the modal state string digit value, endorsing those relationships (or models) accordingly.

6.2 Independent action task environment

A further dimension of complexity is supported by the action task environments in that the off-diagonal, non-zero elements of the action-state change matrix (ASCM) can be increased as well as increasing the non-zero elements of the SSCM. In order cleanly to compare the effects of complexity on the results obtained in the recognition task environment with the results from the independent action task environment, increasing SSCM-based complexity alone will be undertaken first and then, for given levels of SSCM-based complexity, different levels of ASCM-based complexity will be modelled.

In keeping with the specification of the Moss (1998) model, the objective of the organization will be to maintain a set of target values of observed digits in the state string. In these experiments, the initial state string provides the target values for the organization. The state string is then perturbed by mutation and the task of the organization is to bring the value of the state string back to its target value. If all events are to be “off”, then the natural initial (and target) value of the state string is 0 at every position.

6.3 Co-operative task environment

Two sets of experiments are required here. The first is to replicate the task environment of the VDT model itself and the second to compare results in the cooperative action task environment with the results obtained in experiments with the other task environments.

The purpose of the VDT model was to represent a design problem. That is, the relevant “environment” was, for example, a space launch vehicle which did not exist and the task was to create a vehicle with specified characteristics. In effect, the space launch vehicle had no characteristics at the start of the design process and had a specified set of requisite characteristics at the end. If we consider each digit in the state string to represent the absence (if 0) or presence (if 1) of a characteristic, then the initial state string would contain all zeroes and the target state string would contain all 1s and the actions would change the state string over the course of the simulation from the initial to the target string. The first set of experiments demonstrate that the specification of cognition and the process of organization restructuring implemented in this model supports the achievement of all design objectives. One question of interest here is how robustly effective the organizational structuring process is with respect to the complexity of inter-relationships among the design features (SSCM-based complexity).

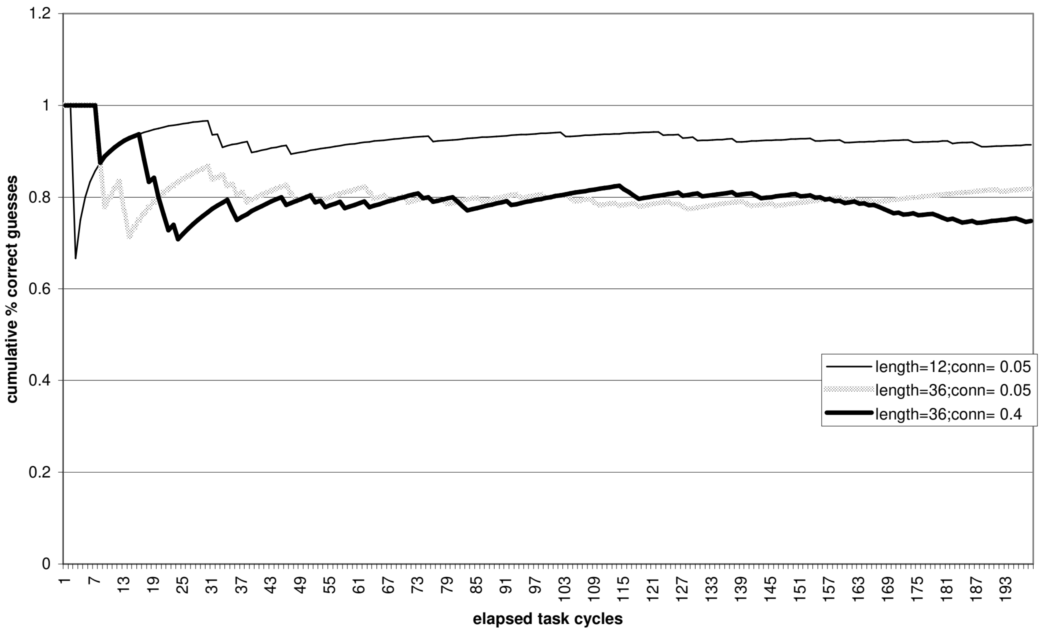
The second set of experiments relates the co-operative action task environment back to the previously considered environments by taking the target state to be the initial state with random perturbations. The effect of both types of complexity in the cooperative action environment can then be compared with the effects of complexity in the independent action task environment.

7 Experimental results

7.1 Recognition task environment

The recognition task environment model was run with three configurations in which different state string lengths and different degrees of connectivity among states were set. The agents were adaptive but no changes in organizational structure or reporting arrangements were allowed. The cumulative percentage of correct assessments of the modal digit value by the top-most manager is reported in Figure 8. These results are representative of the outputs from the runs conducted with these models. There are nine observer-agents who report their views of the modal digit value to their managers who, in turn, report to the top-most manager. If the state string is of length nine, then as Carley and Svoboda pointed out, a majority voting rule will correctly identify the modal digit value. However the rule used by managers in this model was to take the views of the majority positively endorsed subordinates.

Figure 8: Recognition task environment: results with three configurations



With a nine-digit state string, accuracy was perfect. The 12-digit and 36-digit state strings gave more interesting results and these are reported in Figure 8. By inspection, the effect of the initial conditions ceases to dominate the series by the 50th task cycle. The average cumulative percentage of correct guesses over the remaining 149 task cycles was just over 92.5% while for the two runs with 36-digit state strings, the averages were just under 79% and just over 79.5%. There is no chance in that period of either a type I or a type II error in correctly assigning an observation to simulation run with a 12-digit or a 36-digit state string.

If we take the same interval of task cycles to distinguish between the two runs with 36-digit state strings but different degrees of connectivity among states, then formally the series with the higher degree of connectivity (0.4) has the lower average of correct assessments of the modal digit value than does the series with the lower degree of connectivity (0.05). However this result is spurious since, eliminating the last 30 observations from the series (that is from the point where the two time series cross), the series reflecting the higher degree of connectivity shows the higher average of correct assessments (0.793) than does the series reflecting the lower degree of connectivity (0.798) also with high levels of confidence (at least 99.9% in all cases).

The conjecture that arises naturally from these results is that complexity of relations among the digits of the state string has no unambiguous effect while increasing the length of the state string reduces accuracy of organization judgement. It is simply noted for further investigation that the results with the longer string are remarkably close to the level of the cumulative correct estimates found by Carley and Svoboda for the nine-digit state string at 76.14% for the case where individuals learn but there is no structural change in the organization.

7.2 Independent action task environment

The measure of organizational efficiency used in the models with action task environments was the number of task cycles that elapsed between the time the value of a digit was changed from its target value until the time it was returned to its target value. The results of four simulation runs, collected in Figure 9, indicate that the results are more sensitive to the length of the state string that to the complexity of the relations among the digits of the state string or, in this case, the complexity of relations among actions and state-string digits.

All of the simulation runs were conducted over two hundred event cycles in 50 days. At the start of each day, the CEO could change the manager to which any agent reported but not the fundamental organizational structure.

The forward-most row on the y-axis of the chart in Figure 9 is obtained from a simulation run in which the state string had 12 digits while in all other runs the state string had 36 digits. As indicated along the x-axis, giving the intervals of event durations, a considerably higher proportion of the events in the 12-digit simulation were resolved within two event cycles than in any of the 36-digit simulations including the 36-digit simulation which was otherwise identical to the 12-digit simulation.

In none of the simulation runs were there many episodes lasting more than 10 event cycles although even that number was reduced to zero by either higher degrees of interaction among states (SSCM-connectivity) or between actions and states

(ASCM- connectivity). These effects of SSCM- and ASCM- connectivity are more easily seen in Figure 10.

Figure 9: Distribution of event durations (independent actions)

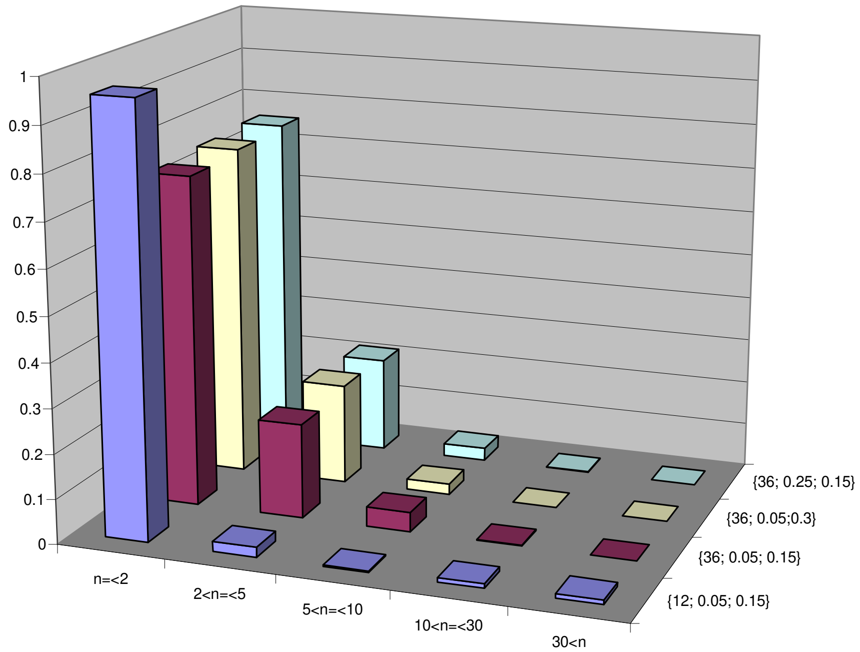
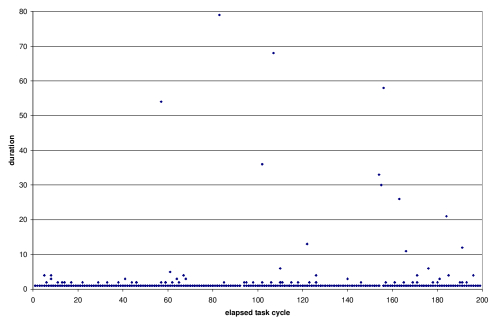
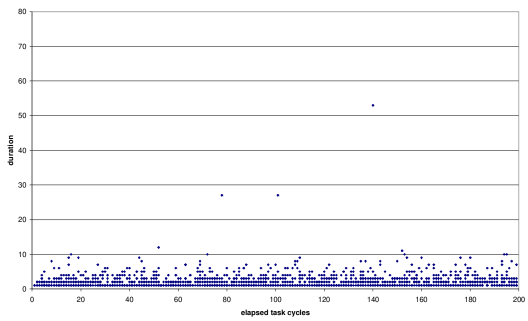
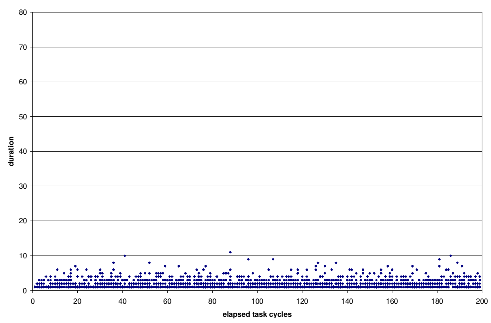
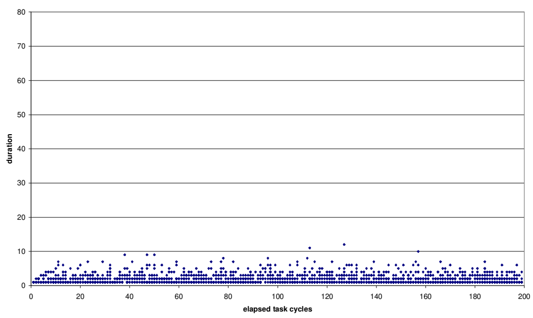


Figure 10: Scatters of event durations over time (independent action)

Length=12; SSCM-conn=0.05; ASCM-conn=0.15	Length=36; SSCM-conn=0.05; ASCM-conn=0.15
	
Length=36; SSCM-conn=0.05; ASCM-conn=0.3	Length=36; SSCM-conn=0.25; ASCM-conn=0.15
	

All of the scatter diagrams in Figure 10 are on the same scales of elapsed task cycles (the x-axis) and event durations (the y-axis). Each point corresponds to the elapsed task cycle at which an event ended (x) and the duration of the event (y). In the top row, are the scatters for two runs with low SSCM- and ASCM-activity. Clearly, although a larger proportion of events are resolved within two event cycles in the 12-digit simulation, there is also a higher volatility of the durations as indicated by the dots scattered up to 80 task cycles as well as their distributions in the upper reaches of the chart through the simulation. The scatter in the upper reaches of of the upper right hand chart is less pronounced, though again it is not related in any way to the passage of time. It appears from the two lower charts that either a higher degree of ASCM-connectivity or of SSCM connectivity provides enough information to the agents and the CEO to forestall the longer durations of events. The longest duration in either case was seven event cycles as opposed to 80 and 60 event cycles, respectively, in the lower-connectivity runs.

The purpose of this paper is not to enter into detailed analysis of the reasons for these changes but, rather, to identify the questions that require more complicated models if they are to be answered by means of simulation experiments. Nonetheless, one possibility for the cause of these differences is that the CEO will have had more information resulting from conflicting actions by agents in the higher connectivity runs and, so, was adjusting the reporting relations among workers and managers more actively. If so, the activities undertaken within the organization and the conditions in which they are undertaken are important influences on managers' ability to modify organizational structure in order to improve organizational performance. This conjecture is entirely consistent with Alfred Chandler's (1962) historical analaysis of the development of, for example, the multi-divisional firm.

7.3 Co-operative action task environment

The experiments with the co-operative action task environment were set up identically to the experiments with the independent action task environment except that the critical path model was specified in addition. Apart from that difference, the effects of which are the subject of interest here, experiments were run with 12- and 36-digit strings and the same parameter values for generating the ASCM and the SSCM. In keeping with the purpose of the VDT model, a series of experiments was run in which the ASCM was the identity matrix of rank equal to the length of the state string and the initial state string contained only 0s while the target state string contained only 1s.

The critical path network was generated by setting

- the maximum path length from an atomic action to a final action that determined the value of a digit in the *a*-string,
- the number of further actions that any action in the network could support and
- the maximum number of supporting actions for each action.

In every experiment, the maximum path length was 7, the maximum number of supported actions by any action was 3 and the maximum number of supporting actions for any action was 4. The network was generated by creating a sump of actions defined by the maximum number of branches from the action, the maximum number of branches to the action and the maximum length of any path to the action. For each action in the sump, each of the three parameters were chosen at random from the interval $[1,m]$ where m was the globally defined maximum number of supported or supporting actions or path length, respectively.

In the test runs, conforming as much as possible to the VDT model, the number of task cycles in which the state string digits were all converted from 0s to 1s was in every case the minimum possible – *i.e.*, the longest path length in the network. This indicated that the representation of cognition implemented in the model used for this series of simulations was efficient in the absence of any ASCM or SSCM complexity.

The equivalent of Figure 9 for the co-operative task environment is Figure 11. While there looks to be greater variability among the results from different configurations of the simulation runs, we observe once again that the experiments with the longer state strings all entail a smaller proportion of the lowest event durations than the experiment with the shorter state string. Of course, in all cases, the durations tended to be a little longer because the shortest possible duration was the length of the longest action path required to effect the relevant final action.

The scatters of event durations over the simulation runs are again in the pattern observed in the independent action task environment experiments.

Figure 11: Distribution of event durations (co-operative actions)

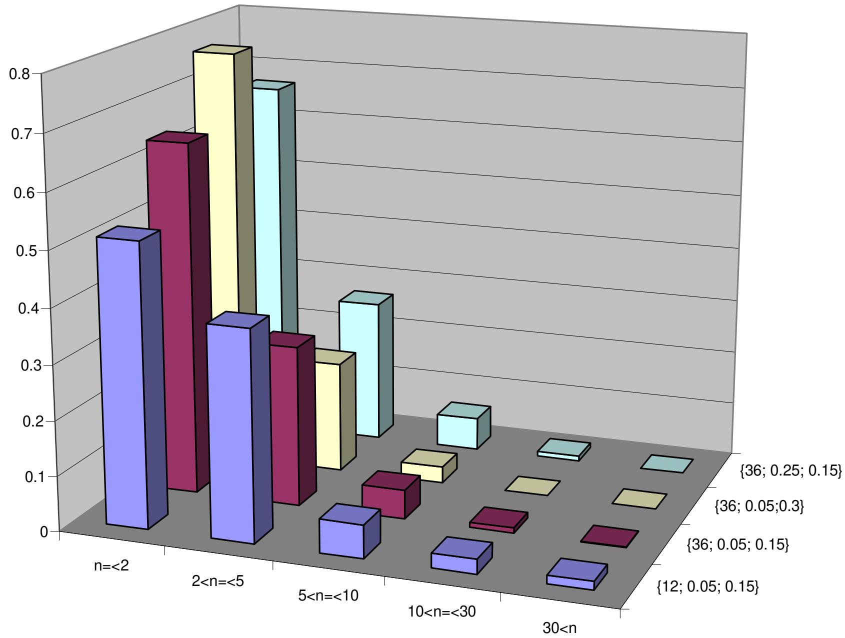
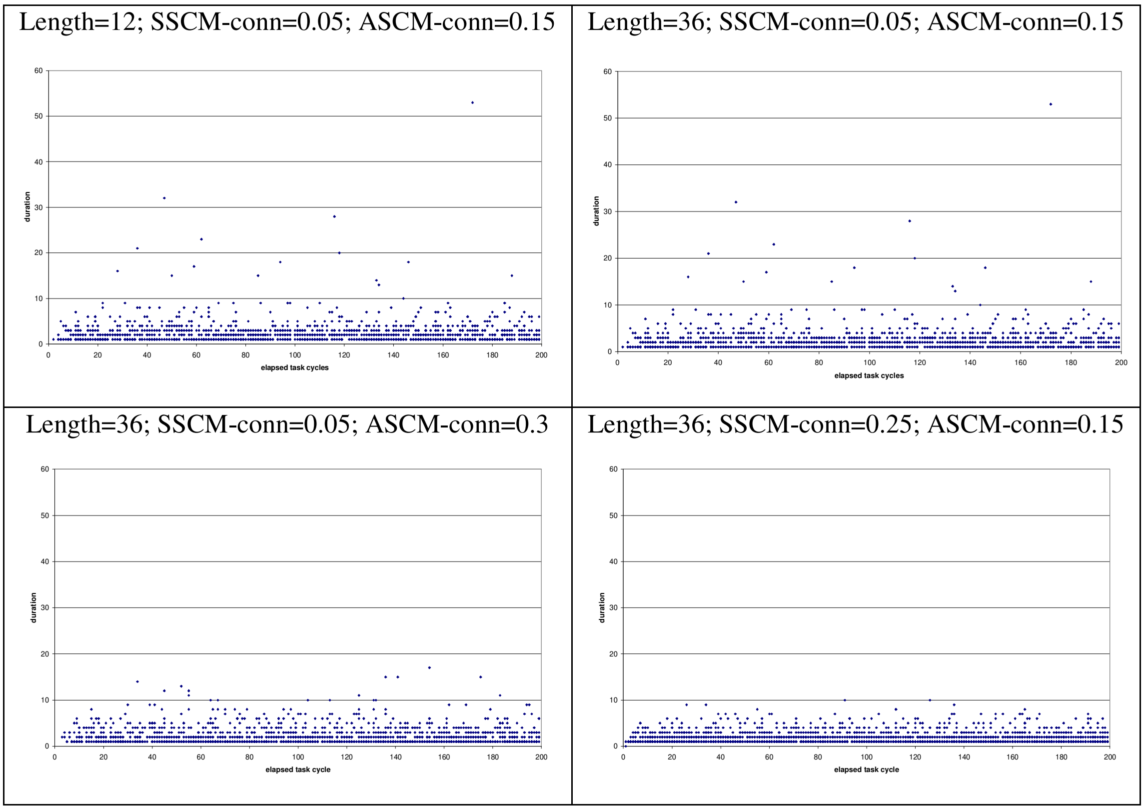


Figure 12: Scatters of event durations over time (co-operative action)



The same pattern of results is apparent in Figure 12 as in Figure 10 insofar as either higher SSCM connectivity or higher ASCM connectivity is associated with the absence of extreme values of the duration of events. It is also clear from a comparison of Figure 11 with Figure 9 that there is more variability among the durations of events corresponding to the various simulation runs with 36-digit state strings in the co-operative activity task environment than in the independent activity task environment.

7.4 Comparison

It is important to recognise that the comparisons made here are not based on significant amounts of experimental data. A few simulations have been run and the reported results are typical of those runs. The purpose remains that of illustrating the role of canonical task environments in specifying questions and establishing a framework that substitutes for closed-discipline theory in relating models to one another and to problem domains.

The clear difference observed here between the recognition task environment and the action task environments is that an increase in state string length reduces the efficiency of an organization in identifying the modal digit value while it increases the efficiency with which agents are able to identify and act on deviations of actual from target digit values. The latter result is found for both independent and cooperative action task environments.

It has already been noted that the effect of connectivity or complexity either between actions and state aspects or among different aspects of the environmental state are associated with an absence of events of long duration whereas the absense of such connectivity is associated with a scattering of events of extreme duration.

8 Conclusion

The results obtained in the reported simulation experiments indicate that agents that can act on their environments are more efficient in richer environments while agents that seek only to recognize a pattern in their environment are less efficient in richer environments. The richer environment gives the agents more information and the ability to act enables them to test their understanding of relationships in that environment. The endorsement schemes used to represent the consequences of experience for agents enabled them to construct and retain models of their environments which were validated by the correctness of their predictions. Consequently, the richer the environment, the finer the relationships that can be identified and the ability to act gave the agents information about changes in the states of their environments as well as information about the states themselves. In effect, action enabled the agents to test relationships among first differences as well as levels while in a recognition task environment they could only observe levels or an analogy thereto.

If this account is correct, it suggests that the Carley-Svoboda and Ye-Carley results are not general. In particular, they do not extend directly to organizations that influence their environments. This is not to suggest that the results on the relative efficiencies of organizational structures will not translate to other task environments but, rather, that those results must be tested independently in simulations of those other task environments.

This is a benefit of the canonical task environment. The differences between the models are clear so that differences in results must be related to those clear and formal differences in the specification of the task environments. It might well be that the specific differences are a consequence of the representation of cognition and that other representations would yield another set of differences in experimental results. If so, then we have a further issue to analyse in the development of our panoply of social simulation techniques and representations. Moreover, these further issues relate in a clear manner to the canonical task environments just as, in closed disciplines, issues relate in clear ways to the theoretical structures that enclose the discipline.

9 Directions for further research

The canonical task environment was implemented to support one feature that was not used in the experiments reported here.

Since agents do not observe the SSCM, they must formulate mental models about the relationships among state string digit values. However, there can be digits that are not observed or observable by agents. In such cases, the columns of the ASCM corresponding to those unobservable digits are themselves unobservable since, otherwise, the agents would know the effects of actions on aspects of the environment of which they are unaware. A consequence of this setup is that some actions taken by agents will have unobservable side effects that, through the SSCM, will influence the digits they can observe. Since these effects can be the result of the actions taken by any agent, there is an inherent variability in pattern of changes in the state string that is not random and yet is not readily predictable by the agents in the models. This increases the difficulty of the mental modelling process and its effects on exiting models would give further indication of the effects of complexity on the models implemented reported here.

A second natural line of development would be to replace the ASCM and the SSCM with more elaborate relationships among actions and states and among different aspects of the state of the environment. For example, there are a number of models of different aspects and grains of climate change. The FUND model [14] has a set of equations for determining global mean temperature (GMT). The exogenous variables of those equations are emissions of greenhouse gases (GHGs) on the basis of which the model returns the GMT. A pilot simulation model relating agent behaviour resulting in GHG emissions and the consequences has been cast in the canonical task environment framework by decoding the action digit strings into rates of change of GHG emissions and encoding the GMT as a segment of a state string. Moreover, recognition of unknown side effects and environmental interaction is being taken into account by augmenting the state string with unobservable digits with an ASCM and a SSCM relating the emissions to unobservable aspects of the climate that influence the observable aspects. This work will be reported in due course and is mentioned here only to indicate directions in which social simulation models that support policy analysis can be integrated into a coherent programme of modelling research without impoverishment of environmental representations.

The limitations of digit strings as a basis for canonical task environments have not been investigated. It is obviously possible that other representations of actions and the environment will turn out to be more appropriate either in particular applications domains or in general. This is an issue for further investigation.

References

[1] Axtell, R., R. Axelrod, J.M. Epstein and M.D. Cohen (1996), “Aligning Simulation Models: A Case Study and Results”, *Computational and Mathematical Organization Theory* 1(2), pp. 123-141.

[2] Binmore, K., M. Piccione and L. Samuelson (1998), “Evolutionary Stability in Alternating-Offers Bargaining Games”, Journal of Economic Theory” 80(2), pp.257-291.

[3] Carley, K. M. and D. Svoboda (1996), "Modeling Organizational Adaptation as a Simulated Annealing Process," *Sociological Methods and Research* 25(1), pp. 138-168.

[4] Cohen, P.R. (1985), Heuristic Reasoning: An Artificial Intelligence Approach (Boston: Pitman Advanced Publishing Program).

[5] Conte, Rosaria, Rainer Hegselmann and Pietro Terna (1997), *Simulating Social Phenomena* (Berlin: Springer-Verlag, Lecture Notes in Economics and Mathematical Systems)

[6] Gilbert G.N. and R. Conte (1995), *Artificial Societies*, (London: UCL Press).

[7] Jin, Y. and R. Levitt (1996), "The Virtual Design Team: A computational Model of Project Organizations", *Computational and Mathematical Organization Theory*, v. 2, pp. 171-195.

[8] Moss, Scott (1998). "Critical Incident Management: An Empirically Derived Computational Model", *Journal of Artificial Societies and Social Simulation*, **1**(4), http://www.soc.surrey.ac.uk/JASSS/1/4/1.html

[9] Moss, Scott and Kerstin Dautenhahn (1998), "Hierarchical Organization of Robots: A Social Simulation Study" (Manchester: Centre for Policy Modelling Technical Report 98-36) < http://www.cpm.mmu.ac.uk/cpmrep36.html>.

[10] Moss, Scott and Esther-Mirjam Sent (1998), "Boundedly *versus* Procedureally Rational Expectations" in Andrew Hughes-Hallett and Peter McAdam (eds), *Analyses in Macro Modelling* (Amsterdam: Kluwer Academic Publishers), in press.

[11] Moss, Scott , Helen Gaylard, Steve Wallis and Bruce Edmonds (1998), SDML: A Multi-Agent Language for Organizational Modelling, *Computational and MathematicalOrganization Theory* **4**, (1), 43-70.

[12] Newell, A.(1990), *Unified Theories of Cognition*, (Cambridge MA: Harvard University Press).

[13] Terna, 1997, "A Laboratory for Agent Based Computational Economics: The Self-development of Consistency in Agents' Behaviour" in [5], pp. 73-88.

[14] Tol, R. S.J. (1996), *A decision-analytic treatise of the enhanced greenhouse effect*, (Amsterdam: Vrije University).

[15] Ye, M. and K.E. Carley (1995), "Radar Soar: towards an artificial organization composed of intelligent agents", *Journal of Mathematical Sociology*, **20**, pp. 219-246.