Validation and Verification of Computational Models with Multiple Cognitive Agents

Scott Moss, Bruce Edmonds and Steve Wallis
Centre for Policy Modelling
Manchester Metropolitan University
Manchester, M1 3GH, UK.
`http://www.cpm.mmu.ac.uk/`

Wider issues of the validation of computational models - ascertaining that they are sound and consistent relative to some logical formalism and/or substantive theory - have not been a subject of the management science literature. In this paper, we demonstrate that computational models can be sound and consistent relative both to a fragment of strongly grounded autoepistemic logic (FOSGAL) and to theories of cognition without losing the expressiveness found in the informally oriented literature on organizational learning and business strategy. Validation is achieved by implementing models and their theoretical components in a programming language which corresponds to a known formal logic. The language used in this paper is SDML. The correspondence of SDML to autoepistemic logic is explained and justified. Issues associated with the verification of models - how well they correspond to observation - are also considered and extended. Benefits of explicit validation and verification of computational models are demonstrated by the implementation in SDML of a computational model of the critical-incident management organization of one of the largest public utilities in Europe. On the basis of the reported simulation results with the model, several research issues are identified both for the development of validation practices in the management sciences and for the analysis of crisis management. contents

1 Introduction

Supporters of the current trends in corporate reorganization emphasize the benefits from downsizing, flattening and local empowerment for the global competitiveness of the enterprise.

Hammer [8], for example, argued that advances in information technology imply a wholesale restructuring of the interactions among the individual processes of the organization. In effect, though information technology has developed to solve one set of problems, the resulting technology has a much wider set of uses and the application of information technology should be directed to the most constraining business processes. The definition of business process is not always stated clearly in the literature but many contributors cite (and presumably do not disagree with) Davenport [6]. He argues that a "process approach... implies a relatively heavy emphasis on improving how work is done, in contrast to a focus on which specific products or services are delivered to customers." In other words, the opportunity costs arise from the processes undertaken within the organization and to focus on the outcomes is to focus on the wrong target.

Formal models of business process change are rejected by leading management theorists such as Weick [25] who argue that a richer language of discourse is required than formal models can provide. However, richness has costs as well as benefits. The current literature reveals many different interpretations of such widely used terms as organizational capabilities and competences. Richness is thus at the cost of imprecision in the language of discourse used to develop key concepts. A consequence of this imprecision is the difficulty of identifying independent tests of the management theories before they are put into practice with the possibility of substantial damage to the implementing organizations.

Evidently, the ability to create formal and testable models which support the same richness of discourse and applicability to (for example) business process analysis as do current approaches to business strategy would be an important advance in management science. The purpose of this paper is to define a development path which would advance the discipline significantly in that direction and to demonstrate the practicality of such a development with a detailed example on a useful scale.

The example we use highlights the difficulties organizations face in dealing with critical incidents from time to time that are not a part of their ongoing, operational routines. In some cases, these incidents can escalate into full-blown crises with catastrophic effects on the organization and its managers. A particularly compelling example of the possible effects of such incidents relates to providers of water and sewage services in the United Kingdom. Incidents involving either significant environmental damage or the supply of contaminated water can lead to gaol sentences for the company's directors in addition to the award of large damages to consumers in the courts.

One such enterprise, which has recently undergone reengineering of its business processes, is the provider of water and sewage services in the north west of England. North West Water PLC merged in 1996 with Norweb PLC, the electricity supplier for northwest England, to form United Utilities PLC. The reorganization associated with the merger changed the remit of operations managers in the regulated water and sewage activities of the company. Whereas operations managers had been expected to handle such incidents as power failures, equipment breakdowns and mains collapses independently before the reorganization, they do not now have the resources to support such independence and are expected to call on centrally provided services to contain and remedy such incidents. Such incidents occur perhaps several times a year in the operations of North West Water and, if not dealt with in a timely and effective fashion, could lead to major crises.

Because the critical incident procedures are safety critical, confidence in the implications of a computational model should not be taken lightly. Verification of the model by comparing its outputs with experience obviously cannot precede the reorganization. Nonetheless, individual components of a model can be verified against relevant experience. Moreover, validation against some well understood formal system to ensure that the model is at least logically consistent and sound would provide a basis for discussion and examination of the assumptions and relationships underlying the model.

In order to demonstrate the feasibility of this approach, a discussion of the essential elements of such validation and verification is provided in section 2 with a discussion of the implications for modelling approaches in section 3. A model of the management by North West Water of critical incident procedures is described in section 4 with the results of simulations with that model reported in section 5. The validation and verification issues associated with the model and results are explored in section 6 followed in section 7 by a discussion of the implications for future research directions.

2 Validation and verification

Validation is the process of ensuring that a computer program will not crash and also that it will perform in the manner intended by its designers and implementors. Verification is the process of assessing the goodness of fit to the characteristics of the models empirical referents. These two topics are considered in turn.

2.1 Validation

If a program runs without error in any computer programming language, then that program is consistent and sound relative to that language. That is, the program does not generate or entail mutually contradictory statements and it does not generate statements which the language does not support.

If the programming language corresponds to a logical formalism, then any program viewed as a set of statements or sentences which runs in that language will necessarily be sound and consistent relative to that logical formalism. One such language, implemented precisely to capture this feature of programs, is SDML [17] which corresponds to a fragment of strongly-grounded auto-epistemic logic [11].

Programming languages generally make it easier to do some things than others. Fortran is optimised for numerical calculation; LISP for functional programming, PROLOG for backward-chaining deduction, and so on. Numerical calculations, functional programming and backward chaining can all be programmed in any of these languages though, as stated, there is a clear correspondence between the ease of programming in any of these styles and the language used.

The immediate advantage of programming in a language which corresponds to a logical formalism is that the properties of that formalism can be understood and proved. Consequently, there are no hidden assumptions in the form of unstated axioms or rules of inference. The particular logical formalism of SDML has emerged as one which supports the kind of multi-agent, strictly declarative modelling favoured by the SDML user community. It is by no means the only possible or appropriate logical formalism for modelling organizations. Indeed, the choice of logical formalisms for different classes of problems is already a fruitful field of enquiry in the artificial intelligence literature. A natural extension of the work described in this paper is the explication of the properties of appropriate logics to underpin a language of discourse for the management sciences.

In SDML, and therefore in the models reported below, each agent is defined on a rulebase and database for each period of time. Every rule in the rulebases and every clause asserted to the databases is sound and consistent relative to strongly grounded auto-epistemic logic.

2.1.1 Strongly grounded auto-epistemic logic (SGAL)

Autoepistemic logics in general were introduced to capture the logic of an ideal reasoner who could reason about his own beliefs. Strongly grounded autoepistemic logic is built on the foundation of the modal logic KD45 with a “belief” operator. The distinction is drawn in this logic between facts which are true (p) and those that are believed to be true ($\Box p$). In particular this differs from normal modal logics in that it is *not* rational to assume that $\Box p \to p$ (that just because you believe something it must be necessarily true).

The semantics of SGAL allow one to reason from one’s own *lack* of belief about something, e.g. *if I don’t believe that I have a sister then I can assume that I do not have a sister.* This is done by allowing a restricted range of assumptions to be introduced and conclusions to be drawn from them, on condition that this does not introduce any incoherence into the conclusions. The mechanism for generating and resolving assumptions was introduced to make SDML efficient as a declarative language which supports forward chaining. This point will be taken up presently. Its import here is that, by making the assumptions mechanism consistent with a restricted subset (or fragment) of SGAL (hence a FOSGAL), we not only get efficient rule handling but also a correspondence of models written in SDML to that FOSGAL.

Since the particular choice of logical formalism has substantive implications, we should be clear about the basis of that formalism. Because formal logics are defined by their axioms and rules of inference, we exhibit these here with explanations for the non-logician.

Strongly grounded autoepistemic logic is built up in three stages: the axiomatization of its base formal logic, the modal logic KD45 with the "belief" operator ($\Box p$), extensions which allow negative assumptions to be handled coherently and some extra conditions to make the logic strongly grounded by eliminating certain circularities of reasoning.

The first stage entails the following axioms[1]:

(1) $\Box (p \to q) \to (\Box p \to \Box q)$ (K)

If you believe that p implies q then if you believe p then you believe q.

(2) $\Box p \to \neg\Box\neg p$ (D)

This is the consistency axiom stating that if you believe p the you do not believe that p is false.

(3) $\Box p \to \Box\Box p$ (4)

If you believe p then you believe that you believe p.

(4) $\neg\Box p \to \Box\neg\Box p$ (5)

If you do not believe p then you believe that you do not believe that p is false.

It also entails the two inference rules of *modus ponens* and necessitation:

(5) if p and $p \to q$ then q (MP)

(6) if p is a theorem of KD45 then so is $\Box p$ (Nec)

This last rule just states that you believe all theorems implied by the logic.

This is given a fixed-point semantics, the second stage, by considering extensions of theories in KD45. An extension, T, is defined:

(7) $T = \{\phi | A' \cup \neg\Box\overline{T}_0 \vdash_{KD45} \phi\}$

where $\neg\Box\overline{T_0} = \{\neg\Box\phi | \phi \in L_0 \wedge \neg(\phi \in T)\}$ (L_0 are the set of *ground* terms in the language, i.e. those without any occurrence of the "belief operator") and A' is the set of sentences $\neg\Box\alpha \vee \Box\beta_1 \vee \Box\beta_2 \vee \ldots \vee \Box\beta_n \vee \omega$ in A such that none of $\beta_1, \ldots, \beta_n$ is contained in T.

What makes the logic strongly grounded is the restriction on the range of assumptions that can be introduced and the conclusions drawn from them. This restriction is defined by (7).

The purpose of allowing only negative ground assumptions which do not occur in the extension is to exclude as much "circular" reasoning as possible. The purpose of introducing the restriction on A to A' is so that the assumptions are treated as rules rather than axioms.

In SDML the belief operator corresponds to "inferred", thus $\neg\Box p \to q$ corresponds to the an SDML rule: "if p is not inferred (by SDML) then infer q". In SDML you are only allowed the box operator in conjunction with negation as notInferred ($\neg\Box p$) – I will write this as ~p for brevity (although ~~p is equivalent in K45 to $\Box p \wedge \neg\Box\bot$, where $\bot$ indicates a contradiction and thus in consistent SDML models (i.e. where $\neg\Box\bot$ is true), ~~p can take the place of $\Box p$).

1. The identifier to the right of each equation is its standard identifier in the logic literature. It is convenient for our purposes to refer to them by equation numbers to the left of the equations.

In SDML, rules are limited to a fragment of this logic because neither ‘not inferred’ nor disjunction operators are allowed in the consequences of rules (thus it is a FOSGAL — a *fragment of* SGAL).

2.1.2 The rationale for the correspondence of SDML to FOSGAL

Encoding negative knowledge is well recognized to be difficult simply because it is not in practice possible to store all the facts that are not true. For this reason, SDML follows the conventional practice of storing only positive knowledge and dealing with negation by allowing rules which have ‘not inferred’ operators in their antecedents.

SDML was designed principally as a forward chaining language because drawing inferences from a set of beliefs and facts and then remembering those inferences by asserting them to a database seemed a more natural representation of agent reasoning than backward chaining in which the implication is stated and then the antecedents evaluated to see if the assertion can be justified.[2] In order to perform forward-chaining efficiently, SDML will fire rules but keep track of any assumptions it had to make on the way. This helps to minimise any back-tracking to try alternative assumptions in the more commonly occurring situations. Thus SDML sometimes needs to make inferences from its own *lack of inference* of certain facts, just as in SGAL one can infer from one’s own lack of belief.

At the same time, we want to restrict reasoning in order to exclude as much circularity as possible. That is, we do *not* want to be able to conclude p from the single rule corresponding to $\Box p \to p$. This consideration alone indicates the choice of either the moderately or the strongly grounded forms of autoepistemic logic. Since SDML rules cannot act as axioms but only as ‘tickets’ to get from one inference to another, the strongly grounded form is the most appropriate.

In summary, SDML was designed on pragmatic grounds for efficient forward-chaining, where one can not explicitly store negative knowledge. For this reason the underlying logic is necessarily less conservative in its inferential power than, say, classical logic. The logic with these properties which is at the same time the most tightly controlled in terms of what can be inferred is SGAL. For reasons of efficiency certain forms of rule are not allowed, resulting in FOSGAL.

2.2 Verification

Verification by comparing model output with statistical data series is too well established to warrant detailed consideration here. Numerical forecasting models have been shown on a number of occasions to be improved by expert intervention ([2], [3], [23]). Integrating statistical forecasting models with rulebases incorporating expert judgement has been shown by Collopy and Armstrong [4] and by Moss, Artis and Ormerod [15] to improve forecasts while making the interventions explicit. Moss *et. al.* [15], in particular included in their system an explanations facility which provided the user with a qualitative account of the reasons for interventions by the rulebase. These qualitative reasons were couched in much the same language as that given by model operators for making similar interventions by hand.

There is therefore some precedent for including qualitatively expressed, domain expertise in models of social or economic processes and verifying the qualitative elements of such models through assessment by domain experts. A further development in this area is to integrate well verified theories from other disciplines into our computational models.

One such theory, used in the model reported below, is Alan Newell’s [18] unified theory of cognition. the theory itself was guided by the requirement to mimic the time required by humans for

2. Backward chaining is also supported by SDML. Its main uses are in recursion, especially on lists, and for using one clause of obvious meaning to replace several clauses in forward-chaining rules.

cognitive acts of varying degrees of complexity. Newell argued that a unified theory of cognition should be implemented as a software architecture (a sort of programming language but restricted to be sound and consistent with the theory and also restricted in its functionality so that it can only represent the specified cognitive relationships). The Soar software architecture [11] is an implementation of the Newell theory and it has been assessed against the performance of subjects in a large number of psychological experiments.

Cooper, Fox, Farringdom and Shallice [5] showed that Soar is not the only possible implementation of the Newell theory. In addition, Ye and Carley [26] found that the full cognitive capabilities of Soar became unusable in even a simple multi-agent model. there are two problems here. One is that Soar becomes computationally expensive in multi-agent settings; the other is that the repetitiveness that Soar requires for learning is not found in those models.

On the other hand, Moss, Gaylard, Wallis and Edmonds [17] found that reimplementing Ye's and Carley's Radar-Soar model in SDML reduced the number of computations and the time required to run the model by two to three *orders of magnitude* while replicating the Radar-Soar results. The increased efficiency of Radar-SDML over Radar-Soar stems from the declarative nature of SDML — itself an issue to be considered in more detail in section 3. A happy by-product of the declarative approach is that actions which take place sequentially in Soar occur in parallel in SDML. But the basic cognitive structure of the SDML implementation of the Soar architecture is the same as the structure in Soar and the results of modelling with the SDML implementation are not significantly different from the results of modelling with the Soar implementation. While systematic testing has not yet been completed, we tentatively claim for SDML implementations of the Soar architecture the experimental validation of Soar itself.

Further verification can be obtained by comparing numerical outputs from simulation models with appropriate statistical data series. Indeed, Moss and Edmonds [16] have shown that qualitative models implemented in SDML (and so consistent and sound relative to strongly-grounded auto-epistemic logic) are naturally parameterized using genetic programming and other search techniques on qualitatively valued variables. The parameterizations are chosen to minimize root-mean squared error of numerically valued variables over a sample data set and are then tested on the hold-out set.

We claim therefore that the verification of computational models with qualitative elements should include empirical tests of the behavioural elements of the models, assessments by domain experts and the well established statistical tests of the model's numerical outputs.

3 Modelling paradigms

Two words which crop up in different contexts of relevance to the present discussion are *procedural* and *declarative*. Procedural knowledge is knowledge about how to do something and this knowledge is held by individuals in a way which does not allow it to be communicated directly to other individuals. Declarative knowledge is knowledge of what is true and can be communicated directly to other individuals. For example, an Englishman may have both procedural and declarative knowledge about the game of cricket. He can explain the rules of the game and describe or show a novice how to stand at the wicket or where to stand if he is to play off-stump or what to do if he is the wicket-keeper or the necessity of keeping the bowling arm straight at the elbow. All of this knowledge is declarative. To hit the ball successfully and place it where the batsman wants the ball to land or to spin-bowl so that the ball hits the ground and bounces so as to hit the wicket without coming into the range of the bat require abilities that can only be attained by practice. However well a person might know the rules and be able to describe the practices of cricket, that person will not be able actually to play cricket without acquiring substantial procedural knowledge. Interestingly, this distinction was also made by Edith Penrose [19] in her seminal (1959) analysis of the direction of the growth of the firm although she called the two types of knowledge objective and subjective. But her distinction between the two was couched in the same terms as Anderson's [1] discussion of the distinction between procedural and declarative knowledge. Discussions of core competencies and capabilities in the business strategy literature are based on the belief that organizations have procedural knowledge which cannot simply be imitated by other organizations.

A similar, though by no means identical, distinction is made by computer scientists between declarative and procedural programming and programming languages. Procedural programming languages include C, Pascal, Fortran and Basic. All of these languages require the programmer to write out the sequence of steps required to fulfil the objectives of the program in the order in which they are required to be completed. Declarative programming languages include Prolog (which also has procedural features) and SDML (which is more strictly declarative). Programming in these languages entails the writing of statements and relationships among statements which are "true" (in the sense of being tautologies). A statement can be represented by clauses on databases and the relationships can be represented as rules. Typically, there are rules in such languages which state that one set of statements is true if some other set of statements is true. A virtue of forward chaining is that statements demonstrated by rules to be true are stored on a database for future retrieval.

These two uses of the declarative-procedural distinction are seen to be mutually reinforcing in relation to computational models and the interpretation of the behavioural elements of these models by social scientists. That is, the distinctions between procedural and declarative programming and between procedural and declarative knowledge, respectively, are closely related in computational modelling.

3.1 Procedural modelling

If a process is specified in advance of its being undertaken, it is in effect an algorithm with ordered steps carried out in some specified, though often conditional, sequence. An economist, for example, would model the demand for particular goods as the result of applying an algorithm for the maximization of a consumer's utility subject to the constraints of known prices and incomes. An operational researcher would specify the steps required to maximize outputs or minimize costs within the constraints imposed by a specified technology, prices and available resources. Economists (*e.g.* [21]) have long denied that individuals actually apply constrained-optimization algorithms to the determination of the goods they consume or the labour they offer. The conventional wisdom among economists in particular is that individuals act *as if* they apply such algorithms in reaching their consumption and labour-supply decisions. In effect, the constrained-optimization algorithms are procedural descriptions (in the programming sense) of procedural knowledge (in the cognitive-science sense) of individual agents in the economy.

It is important to note in this context that procedural modelling applies the description of a specified process in order to determine the state that will emerge. The process itself cannot emerge from the model precisely because it is a well-defined algorithm already embedded in the model.

3.2 Declarative modelling

Any description of the behaviour of an individual or a system in a declarative programming language takes the form of a set of implications. If one set of statements is true (the antecedents) then another set of statements is true (the consequents). The antecedents describe in general terms the state in which the consequents are true. The consequents could describe new characteristics of the state including the results of specific actions undertaken by individuals. In either event, the *antecedent→consequent* coupling describes one step or one set of parallel steps to be taken in a particular state. It is usual for both antecedent and consequent statements to contain variables. The antecedents' variables are unified with particular values either by matching statements already on a database or by inferring values from such unified statements. A standard example is that the antecedent-consequent couple *isa ?x person→loves ?x mary* means every person loves Mary. So the statement on a database *loves john mary* is true by implication if the statement *isa john person* is on the database. There can be a set of such statements which depend on one another including, in this example, an antecedent with the consequent *isa ?x person* that unifies the variable *?x* with *john*.

Because the sequence in which these implications are found to be true for particular instances (*i.e.* the order in which these implications are instantiated) is not imposed *a priori* in the writing of the declarative model, they can be said to emerge by computation with the model. So when we run a declarative simulation model, the sequence of implications which are drawn, including implications that entail actions by individuals, emerges during the course of the run.

Whereas the processes are programmed to determine the resulting states in a procedural model, the states determine the processes in a declarative model.

3.3 Modelling paradigms, validation and application

It is not intended to suggest here that all numerical models are procedural and therefore that all numerical models give rise to emergent states rather than emergent processes. A clear exception here is system dynamics which is numerical and declarative. Similarly, it is always possible to program declarative models in procedural languages and procedural models in declarative languages. Often, there is some combination of procedural and declarative elements in a single model.

Because declarative languages entail the representation of declarative models as compositions of logic-like rules, it is easier to relate such models and languages to formal logics and to use those logic-like structures to model emergent processes than by using procedural languages and modelling approaches.

The emergent nature of the behaviour makes the declarative modelling approach appropriate for cognitive modelling where the objective is to understand mental processes and responses to various stimuli. As we shall see presently, cognitive modelling can be applied to abstract agents in simulated organizations to capture some aspects of organizational learning as described by, for example, Levitt and March [13] or Huber [9].

In general terms, declarative modelling within a declarative programming environment supports the development of models with strong qualitative elements and a clear correspondence to formal logic for purposes of modelling emergent processes. Such modelling is especially apt where the forces for change are either well understood but their consequences are not or, alternatively, the forces for change are themselves the objects of the analysis. These are precisely the kinds of issues which have concerned scholars such as Prahalad and Bettis [20] who have been concerned with business strategy. Their identification of “dominant logic” within management teams is reflected in the formal logical basis associated with declarative modelling.

Procedural modelling within a procedural programming environment supports models where the processes of change are not of much interest but the achievement or properties of some steady state is of much more interest. It is therefore appropriate for economic analyses where the objective is to demonstrate the properties of equilibrium states or for operational research where the objective is to define the best feasible result.

Declarative models running in a declarative programming environment are most easily validated by demonstrating their consistency and soundness relative to a logical formalism to which the language corresponds and possibly by proving theorems about the properties of the models within that logical formalism. Procedural models and environments would naturally correspond to mathematical formalisms and would naturally be validated in relation to them.

4 An example: the North West Water incident management organization

While the preceding discussion has been concerned with issues at a high level of abstraction, they do have substantive and immediate implications for applied, computational modelling. In this section and section 5, we consider a model implemented in SDML and applied to the analysis of learning in an actual organizational context.

The operations of North West Water entail local empowerment of the operations managers who have to rely on central services and systems for the provision of resources to manage critical incidents. A central systems organization decides on the activities to be undertaken during the course of a critical incident but leaves the planning and scheduling of these activities to a specialist planning and scheduling department. The operational activities modelled here are inspection, reporting and repair. There is an infrastructure of telemetry from operations sites and verbal communications channels among specified agencies within North West Water. The public can report incidents at any time of the day or night but the company's contact point with the public varies over the course of the day. From 8h00 until 22h00 the company's Customer Service Centre takes all calls from the public and passes on calls indicating the possibility of a critical incident directly to Central Systems. During the night-time hours, 22h00 until 8h00, calls from the public are received by the Operations Control Centre who pass on any reports indicating critical incidents to Central Systems. The information flows involved in the incident management procedures are captured in Figure 1.

In this section, we describe first the model structure and the substantive implications of the implementation in SDML and then the representation of cognition including its roots in the Soar and ACT-R programming architectures.

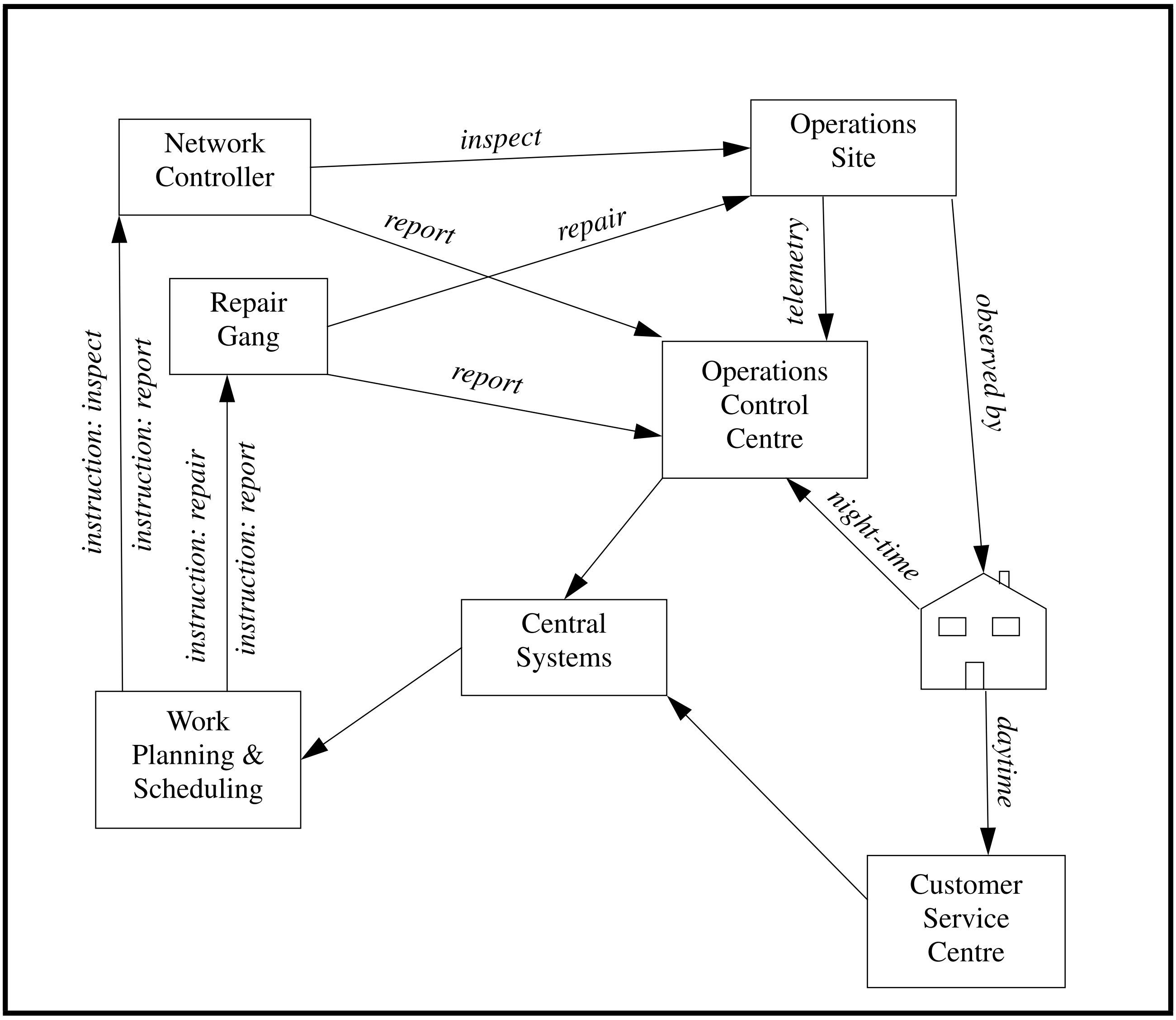

Figure 1: Incident management organization

4.1 NWW model implementation

A key element in any model of an organization must be the specification of who knows what. Some knowledge is common to everyone in the organization, some is common within departments and some only among specific groupings of individuals. It is not always necessary explicitly to model how information comes to be commonly held in this way. Sometimes, however, individuals decide whom to tell about specific phenomena and then address the relevant information to those specified individuals. Such communication can usefully be modelled.

SDML supports a “container hierarchy” in which some types of agents (composite agents) can contain subagents. An organization can contain departments which can, in turn, contain activities. SDML also supports compilation based on the prior definition of clauses which can be asserted to databases. These clauses are defined as part of the definitions of agents. If an agent, say an individual contributing to an activity, is represented as a subagent of the activity and a clause is defined at the level of the activity, then every time the agent asserts an instance of that clause definition, the clause is actually asserted to a database of the activity and can therefore be accessed by every individual in that activity as if it were on their own respective databases. If the clause were defined at the level of the department, then whether asserted by the department, a component activity or an individual contained by an activity, the clause would be stored on the department's database and could be read by every activity and individual as if it were on their own respective databases.[3]

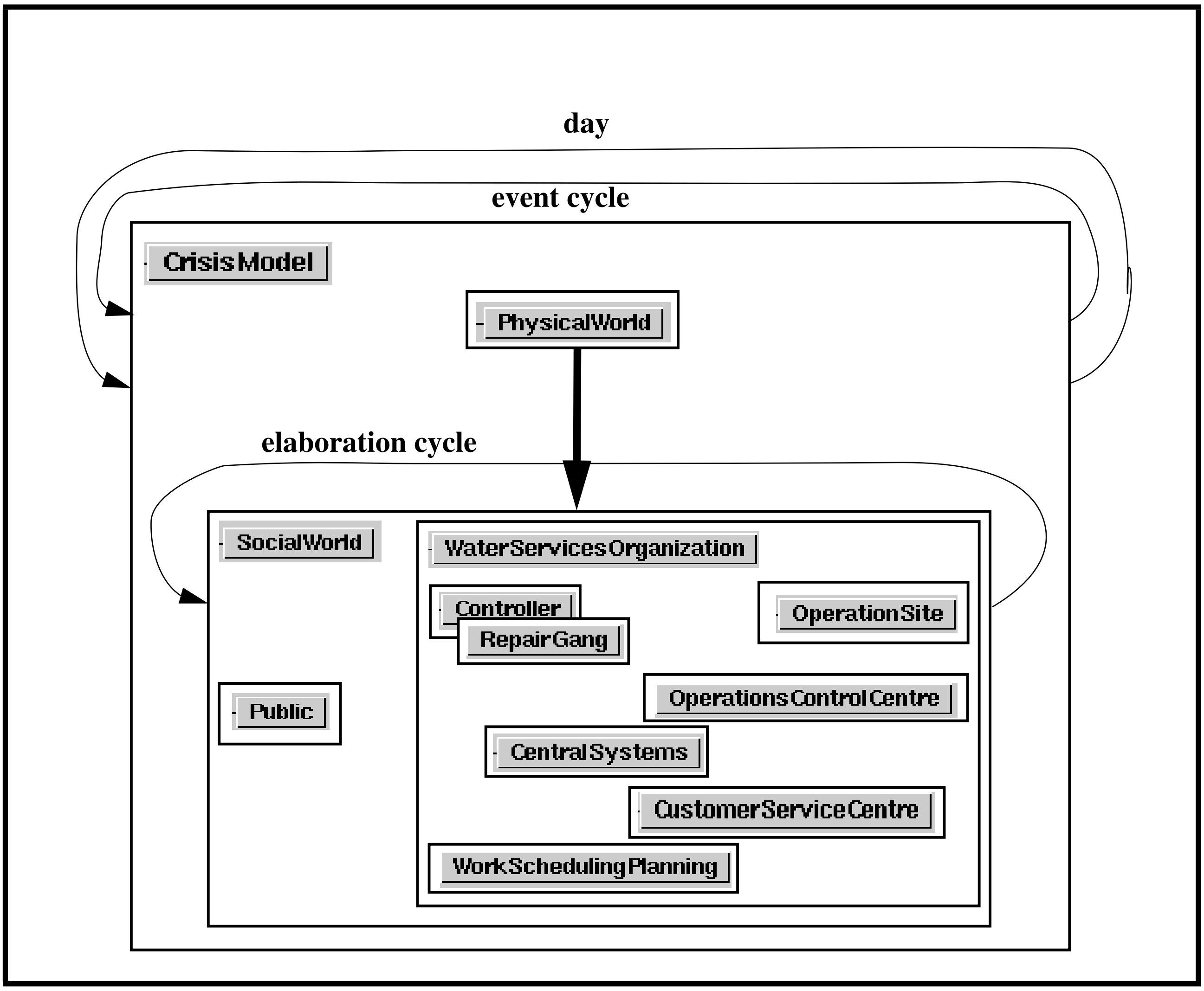

Figure 2: Container structure of critical-incident model

The container structure of the critical-incident model is shown in Figure 2. The outermost container is an instance of *CrisisModel* which specifies the sequence in which its immediate subagents will fire their rulebases.

An instance of the type *PhysicalWorld* has databases with information about the causes and effects of events which can occur at the various operating sites together with the combinations of actions that will remedy an untoward event. Seventeen such events were considered including discoloured, evil tasting or smelling water, road collapse, burst mains or foul sewers, fire, intruders,

3. It should be noted that clause definitions can be made private in which case they can be asserted and retrieved only by the agent on whose databases they are stored. Any such agent must be of the type for which the clause is defined.

contamination incidents, power supply or pump failures and chlorine leaks. These and the other events were allocated probabilities of spontaneous occurrence as well as probabilities of occurring as a consequence of one of the other events. In some cases, the probability of spontaneous occurrence is 0 because the events can only be consequences of some other event. Pollution incidents, for example, are always caused by something. That one event will be a consequence of another involves both a primary and a secondary probability. The primary probability is that the first event causes the second. The secondary probability is that the remedial action in respect of the first event causes the second.

On each occasion when an event might occur, the PhysicalWorld decides which, if any, primary events will occur spontaneously according to the specified probabilities and, if any such event does occur, assigns it to an operating site at random. If there are already events occurring at an operating site, the PhysicalWorld propagates consequential events at random from the specified probabilities and assigns them to the appropriate operations site.

The causes of specific events are not in practice known to the individuals involved in critical incident management until the manifestations of the incident have been observed and the situation analysed. Even then, they might make the wrong diagnosis. For these reasons, it would not be appropriate to post the causes of a particular incident to a database accessible by all agents. Consequently, the relevant clauses are asserted privately by the PhysicalWorld to its own databases and the fact of the occurrence of the event is asserted to the database of the operation site at which the event is to occur. This assertion is achieved by the explicit addressing of the clause *eventOccuring event* where *event* is actually *fire*, *pumpFailure*, *contaminationIncident*, or the like. Once one such event has been allocated to the operating site, then with the appropriate probability all of the consequential events and their consequential events, *etc*. are also allocated at random to that site. In subsequent time frames, secondary consequences are allocated to that site with the specified probabilities whilstever events with those secondary consequences continue. Events, once allocated, remain a feature of the site until they are remedied, if there are remedies, and the events which gave rise to them have been eliminated.

The operating sites (in practice unstaffed) recognize two kinds of event: telemetered and publicly observable events. When a site has had a telemetered event asserted to its databases, it sends a message stating that it has that event to the OperationsControlCentre. When a site has a publicly observable event asserted to its databases, it selects at random a percentage of households and asserts the occurrence of that event to their databases. In the simulations reported here, each household of 100 had a 10 per cent probability of being selected to receive such a message. Because the information contained in those assertions refers to actual information which is available selectively, once again explicit addressing of the assertions is appropriate.

The OperationsControlCentre agent forwards the telemetry and public reports to the CentralSystems agent who decides on the actions to be taken. The instructions to take these actions are addressed explicitly to the WorkPlanningAndScheduling agent who allocates the work by addressing the instructions of an agent of type RepairGang or Controller as appropriate. The reports by the repair gangs or controllers are addressed to CentralSystems agent. Repairs take the form of actions asserted by the repair gang to the operating site and then read from the operating site's databases by the PhysicalWorld instance.

The cognitive behaviour in this model is by the instances of type Controller, and the workPlanningAndfScheduling, OperationControl and CentralSystems instances. These agents learn by specifying and testing models which are held on their respective databases as private clauses — *i.e.* clauses which can be asserted by an agent only to its own database and read only by that agent.

4.2 Agent cognition: learning as modelling

Agent cognition is represented as a process of model generation and testing within a means-ends framework. This approach has its origins in the cognitive science literature, classically Soar as discussed in section 2.2 and, in a slightly different vein, Anderson's [1] ACT-R. Both rely on problem-space architectures which are in effect relationships between goals and the sub-goals needed to achieve those goals. Newell's unified theory of cognition is intended in principle to bring together in a single theory representations of decision-making, learning, memory and other aspects of less relevance here. The ACT-R theory is intended as a theory of memory. Both embody their theoretical structures in software architectures, though Cooper *et. al.* [5] have argued that the theories are not implementation-specific and, so, the Soar and ACT-R software must entail implicit non-theoretical elements.

Both Soar and ACT-R represent cognition as the building up of structures of declarative relationships. Both entail the learning of increasingly complex models as a process of "chunking" though "chunking" itself has different meanings in the two theories. In Soar, chunking amounts to the replacement of procedural knowledge in the form of elements of the problem-space architecture with declarative knowledge in the form of condition-action rules. In ACT-R, chunking is the creation of data structures with special slots. Since some of the slots can themselves contain chunks, there is no theoretical limit on chunk complexity.

There are, as yet, no models of multi-agent interaction in ACT-R. There are several such models implemented in Soar (*e.g.* [26], [22]) but these have a small number of agents and, if any hierarchical relations, only two layers. For the reasons given in 2.2, none allow for chunking. Those reasons do not apply to SDML models because the greater speed of execution allows for events to recur. However, in the North West Water model, the complexity and probabalistic nature of the causal relationships makes exact repetition of some events unlikely and certainly infrequent. So learning dependent on the observation of repetitive events is not a promising representation of cognition in those cases. This result is itself important since it enables us to distinguish between events which, by virtue of organizational learning, become routine and which require special structures, measures and cognitive processes to be embodied in crisis management teams.

In the North West Water model, the controllers build models relating remediable causes to consequences. They are assumed to know which individual events are causes of which other individual events but not the associated probabilities. Because of the differing probabilities of some events occurring as direct and immediate results of other events and some occurring as an indirect consequence and at subsequent times, it is not always clear which causal events are the most important to remedy first. The procedure they follow to formulate models is, in the absence of any applicable model, to generate a new model which postulates as a single cause the kind of event which is a cause of the largest number of the other observed events at the same site. For example, early in one simulation run, two events occurred spontaneously at one of the operating sites: an intruder forced entry and the water pressure dropped. As a result, virtually everything else that could happen did happen. These events included a fire, a chlorine leak, a power supply failure, discoloured water, contamination and pollution, low water levels, no water to customers and a water taste or odour. The controller sent to inspect the site concluded that the key event to resolve was the presence of the intruder because among the observed events, more of them had intrusion as a possible cause than they had for any other causal event. The runner-up as the key event was fire which came second to intrusion only because one possible cause of intrusion is a fire but fires do not cause intrusion.

The models were used by the controllers to identify and report to central systems the primary cause or causes of an incident. If, as a result of that report, the remedy applied on the instruction of central systems eliminated at least one of the events identified by the model, then the model was endorsed as having reduced the severity of the incident. If the result of applying the model was to eliminate all of the events covered by the model (*i.e.* all of the causes and all of the effects), then there was a further endorsement to that effect.

4.2.1 Basic model-specification process

In the first of the simulation setups reported here, each endorsement added one unit of "weight" to the model. Since each model started with a weight of 1 (with an endorsement as a new model), a model which had once been believed to have reduced the number of events at a site had a weight of 2 and, if all of the causes ascribed by the model as the sources of all of the effects and all of the claimed effects of those causes were eliminated, the model had a weight of 3.

Any cognitive agent in the model will try to specialize and to generalize successful models. Specialization involves taking the union of the set of causes specified by two models and the union of the predicted effects. Generalization involves taking the intersection of the causes of two models and the intersection of their predicted effects. Because a specialized model has more conditions of application, it will apply to fewer cases while a generalized model has fewer conditions of application and so is more widely applicable. The specialization and generalization procedures serve the same role as chunking in Soar and ACT-R in that they allow for more complicated patterns and sets of causal relations to be used in the course of cognition. The controller would report every condition of the successful model and would be able to identify with greater accuracy the core causes of any incident. Consequently, we would expect more remedies to be applied more quickly as a result of model specialization and the most effective remedies to be applied as a result of model generalization.

The controllers filtered the models they retained actively in memory over time be remembering from one day to the next only those models with weights of 2 or more.

The controller would select a model to apply by devising a list of candidate models and then choosing the best of the candidates. Among those models which had a single causal element, the agent would consider those which specified as a cause an event which had actually occurred and among all such models would keep in consideration those which had the largest number of actual events among their predicted states. There could be several of these. A multi-cause model would be selected whenever all of the events it treated as conditions and as actions were actually observed.

All of the candidate models were collected and given a probability of being chosen which was proportional to their respective weights. The selected model was then chosen at random according to the weighted probabilities.

4.2.2 Current model-specification process

In the second simulation setup reported here, the basic cognitive process was altered in several ways.

From the perspective of organizational science, the most important addition was that, at the end of each shift, the controller would "tell" the controller on the next shift about any models which had been applied and resulted in the elimination of all hypothesized causes and effects. If the succeeeding controller did not have an equivalent model, it would formulate one. If it already had such a model, it would note the reported success acvhieved with that model.

The remaining changes concerned the valuation of models by the individual controllers. Specifically, the number of endorsements which the controllers could apply to their respective models was increased so that specialized models were valued for their specialization and different endorsements were valued differently. The endorsements were put into classes so that a the "new model" endorsement was in class 0, the "reduced critical events" endorsement was in class 1, the endorsement corresponding to the elimination of all modelled events, the endorsement corresponding to a report of successful use of a model by another controller were both in class 2 and the "specialized model" endorsement was in class 3. The value of an endorsement in each class was 1.2 times the value of the endorsement in the next lower class with the least valuable endorsements (in class 0) having a value of 1.

5 Results

The model results are reported in terms of overall organizational performance as well as an account of the models and procedures which emerged from agents' cognition. Two results have proved to be general relative to these simulations. One is that, the sharing of successful models by controllers is sufficient and essential to the achievement of significant reductions in the time required to run simulations of any given number of time frames. The increased simulation speed implies that the cognitive burden on the agents was much less than in the absence of model-sharing. The second result is that, with model-sharing and increased value accorded to specialized models, the controllers more rapidly and systematically developed procedures for dealing with relatively small incidents but in neither case were the controllers able to formulate procedures for dealing with the larger and more complex sets of critical events.

5.1 Organizational performance

Incidents are better managed if critical events are remedied within a shorter time and have fewer consequential results. In this model the second criterion is a consequence of the first since consequential results occur with a constant likelihood in each event cycle preceding by a causal event. We observed no long-term improvement in organizational performance over any simulation run under any set of conditions. As indicated by the data reported in table 1, where there is learning, it takes place early in the simulation run and the results thereafter show some variance due to environmental noise resulting in different patterns of spontaneous occurrence of events.

The data in Figure 2 is taken from a run of 40 simulation days, each containing 18 event cycles. The vertical axis measures the number of event cycles during which an operating site was continuously host to at least one critical incident. The horizontal axis indicates the date at which the site became incident-free. By observation, the pattern gets thinner from the bottom upwards. That is, the number of incidents resolved in a single event cycle exceeded the number resolved in two or more event cycle; the number solved in two event cycles was greater than the number solved in three or more cycles, and so on. But the pattern from left to right — that is, over the course of the simulation run — shows no systematic changes. Indeed, the longest-lived incident was resolved at elapsed event cycle 552 (day 30, event cycle 12) out of 720 event cycles (40 days) in all. However the density of points towards the bottom of the graph indicates that, most of the time, critical episodes lasted only one or two event cycles.

Table 1:Percentage distributions of episode lengths (no model-sharing)

elapsed event cycles	0 < n ≤ 2	2 < n ≤ 4	4 < n ≤ 6	6 < n ≤ 8	8 < n
0-53	68.75	6.25	12.5	12.5	0
54-107	58.82	17.65	11.76	0	11.76
108-161	60.00	13.33	13.33	6.67	6.67
162-215	40.00	20.00	26.67	13.33	0
216-269	25.00	25.00	25.00	16.67	8.33
270-323	22.22	22.22	33.33	11.11	11.11
324-377	61.11	22.22	16.67	0	0
378-431	53.33	26.67	13.33	6.67	0
432-485	64.71	17.65	5.88	0	11.76
486-539	43.75	25.00	25.00	6.25	0
Avg overall	49.77	19.60	18.35	7.32	4.95
std dev	15.54	5.89	8.18	5.72	5.15

Table 2:Percentage distributions of episode lengths (with model-sharing)

elapsed event cycles	0 < n ≤ 2	2 < n ≤ 4	4 < n ≤ 6	6 < n ≤ 8	8 < n
0-53	37.50	12.50	12.50	37.50	0
54-107	56.26	18.75	12.50	6.25	6.25
108-161	53.33	13.33	0	20.00	13.33
162-215	64.71	29.41	0	5.88	0
216-269	57.89	15.79	21.05	0	5.26
270-323	53.85	23.08	15.38	0	7.69
324-377	68.72	10.53	10.53	0	10.53
378-431	87.50	6.25	0	6.25	0
432-485	50.00	31.25	6.25	0	12.502
486-539	64.71	17.65	11.76	0	5.88
avg overall	59.45	17.85	9.00	7.59	6.14
std dev	12.57	7.64	6.86	11.59	4.77

In table 1 and table 1 we have the distribution over time of lengths of critical incidents with and without model-sharing. Much of the variation in the distribution over the simulation runs is due to the variation in the numbers and types of events occurring within each period. As we see from table 1, after the first 50 or so event cycles, between 50 and 87.5 percent of incidents in every three-day period were completely resolved within two event cycles. The position, as indicated in table 1, was much more variable and generally less successful when agents did not share successful models.

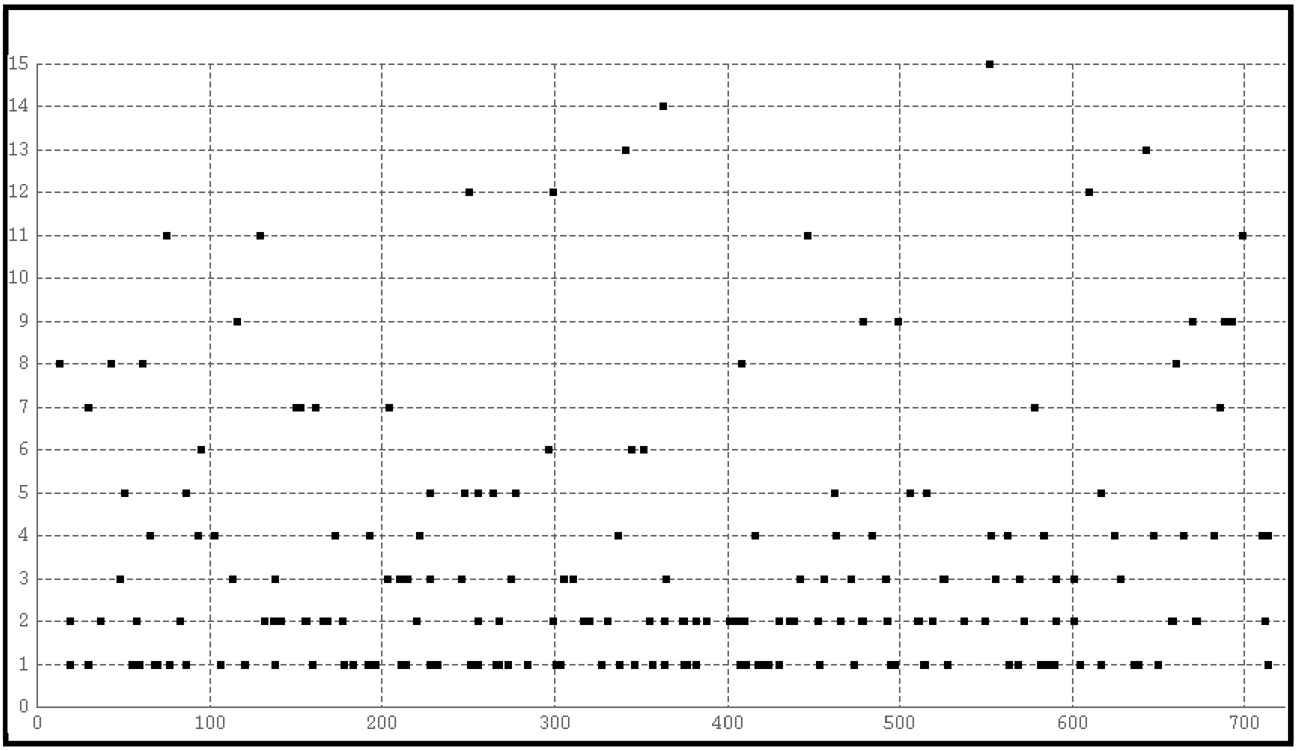

Figure 3: Time pattern of the duration of critical incidents

The difference is in the way that the cognitive development of the controller agents supported the emergence of operating procedures for identifying and dealing with the causes of incidents. Initially, there was no systematic difference. The procedures which emerged to resolve the events *lowLevel*, *lowPressure* and *noWater discolouredWater*, *contaminationOrPollutionIncident* and *tasteOrOdorOfWater* followed from the same cognitive processes in both simulation setups. The excerpt from the simulation transcript reported in table 4 shows how these procedures emerged as a result of the formulation of mental models by the controllers.

The excerpt from the simulation transcript in reports the application by controller-2 of that agent's model designated *controller-2: model-2*. That model specified *lowLevel* as a cause with *lowPressure* and *noWater* as the effects. Since the action *repairLeak* specified by central systems resolves the *lowLevel* problem and there are no other causes of either *lowPressure* or *noWater* at the site, those effects of *lowLevel* are also resolved. This means that all three events related by the model have been eliminated and the model is, as a result, strongly endorsed. The same pattern recurred whenever there was a spontaneous occurrence of *lowLevel*. In addition, the controller with the immediately preceding shift to that of controller-2 repeatedly reported that its own equivalent model had correctly predicted the causal relations. These different sources of endorsement were asserted sufficiently often that *controller-2: model-2* became the second-best endorsed of controller-2's models.

```
Day 0, Event Cycle 5: Spontaneous occurrence of lowLevel at operationSite-3
Day 0, Event Cycle 5: Occurrence of lowPressure at operationSite-3 is an impact consequence of lowLevel
Day 0, Event Cycle 5: Occurrence of noWater at operationSite-3 is an impact consequence of lowLevel
Day 0, Event Cycle 5: Occurrence of discolouredWater at operationSite-3 is an impact consequence of
   noWater
Day 0, Event Cycle 5:Occurrence of lowPressure at operationSite-3 is an impact consequence of noWater
Day 0, Event Cycle 5:Occurrence of contaminationOrPollutionIncident at operationSite-3 is an impact
   consequence of discolouredWater
Day 0, Event Cycle 5:Occurrence of tasteOrOdorOfWater at operationSite-3 is an impact consequence of
   discolouredWater
Day 0, Event Cycle 5:Occurrence of discolouredWater at operationSite-3 is an impact consequence of
   contaminationOrPollutionIncident
centralSystems is instructing inspection of operationSite-3
controller-2 is using controller-2: model-2
      conditions: [(eventOccurring lowLevel)]
      consequences: [(eventOccurring lowPressure) (eventOccurring noWater)]
controller-2is reporting belief ['operationSite-3' (eventOccurring lowLevel)] to centralSystems
centralSystems is instructing remedial action [repairLeak] at operationSite-3
Day 0 Event Cycle 5
      The events at operationSite-3 are [lowPressure discolouredWater lowLevel noWater
   contaminationOrPollutionIncident tasteOrOdorOfWater]
      The remedial actions taken at day 0 event cycle 5 (the previous event cycle) were [repairLeak]
      The events eliminated were [lowPressure lowLevel noWater]
```

Figure 4: Excerpt from simulation transcript (spontaneous occurrence of lowLevel at day 0)

```
Day 8, Event Cycle 15:Spontaneous occurrence of lowLevel at operationSite-1
Day 8, Event Cycle 15:Occurrence of noWater at operationSite-1 is an impact consequence of lowLevel
Day 8, Event Cycle 15:Occurrence of discolouredWater at operationSite-1 is an impact consequence of
   noWater
Day 8, Event Cycle 15:Occurrence of lowPressure at operationSite-1 is an impact consequence of
   noWater
centralSystems is instructing inspection of operationSite-1
controller-3 is using controller-1(reported): model-1
      conditions: [(eventOccurring lowLevel)]
      consequences: [(eventOccurring lowPressure) (eventOccurring noWater)]
controller-3 is reporting belief ['operationSite-1' (eventOccurring lowLevel)] to centralSystems
centralSystems is instructing remedial action [repairLeak] at operationSite-1
Day 8Event Cycle 15
      The events at operationSite-1 are [lowPressure lowLevel noWater discolouredWater]
      The remedial actions taken at day 8 event cycle 15 (the previous event cycle) were [repairLeak]
      The events eliminated were [lowPressure lowLevel noWater discolouredWater]
```

Figure 5: Excerpt from simulation transcript (spontaneous occurrence of lowLevel at day 8)

The same model was adopted by controller-1 on the basis of the report by controller-2. controller-1 then had the same successful experience with it which was reported to, and adopted by the controller in the next shift, controller-3. In the transcript excerpt of its first use, reported for day 8 in Figure 4, the model is identified as *controller-1(reported): model-1*.

The same model, formulated initially by controller-2, was the third best-endorsed model of the other two controllers.The identification of robust and simple relationships such as those associated with the occurrence of the lowLevel event are not necessarily a benefit. Indeed, they sometimes prevented the controller agents from developing more useful models for application in more complicated and difficult episodes. One such episode is reported in the transcript excerpt of Figure 4.

In Figure 4, the episode starts with a spontaneous occurrence of the event *fire* that has a number of immediate and then, in the subsequent event cycle, secondary consequences. The impact of the fire is to create the events *disinfectionFailure*, *lowLevel*, *pumpFailure*, *contaminationOrPollutionIncident* and *noWater.* These impact consequences themselves have impact consequences which, in this case, constitute additional causes of events resulting directly from the fire. controller-1's inspections leads that agent to apply the previously endorsed model *controller-1: model-10* which is itself a specialized model relating *lowLevel* and *contaminationOrPollutionIncident* to the remaining events other than *fire* (and incorrectly including *discolouredWater*). Since *fire* is directly or indirectly a cause of all of the events, no event is remedied until the following event cycle when the model *controller-1: model-3* relating *fire* to other events is applied. In the meantime, however, further secondary consequences of the original *fire* event and its impact effects are manifest which will require further action keeping the incident alive for several more event cycles.

The result of this complexity of actual cause-effect relations relative to the cognitive abilities of the controller agents is that no strategy to reduce the time required to remedy fire-induced episodes has ever been learned in the course of some 30 simulation runs with different assumptions about model sharing and model development.

In part, this failure of effective procedures to emerge for such cases is a result of the limitations of the cognitive behaviour ascribed to the cognitive agents in these simulation models. One natural extension, for example, would be to write rules for cognitive agents to identify sequences of mental models which were successful in eliminating events that became more complicated over time as a result of delayed consequences of earlier events or additional spontaneously generated events. Whether the controllers or some other agent should be given this responsibility is a matter for further experimentation and the results will relate clearly to the crisis management literature. The issues involved will become clear when we have analysed the cognitive activities of the individual controllers in these simulations.

5.2 Individual abilities and performance

Our assessment of the abilities of individual agents is taken from cognitive science. An individual is better able to function effectively in a domain of activity the better that individual "understands" the relationships in that domain. A "better understanding" connotes the recognition of which relationships are applicable or inapplicable and the ability to act more quickly because more complicated relationships are seen and used to determine successful activity in any given situation. At the same time, relationships are not more appropriate in any sense simply because they are more complicated. Indeed, one aspect of a "better understanding" is the choice of the right relationships of the degree of complexity necessary for correct action.

```
Day 4, Event Cycle 9:Spontaneous occurrence of fire at operationSite-1
Day 4, Event Cycle 9:Occurrence of disinfectionFailure at operationSite-1 is an impact consequence of fire
Day 4, Event Cycle 9:Occurrence of lowLevel at operationSite-1 is an impact consequence of fire
Day 4, Event Cycle 9:Occurrence of pumpFailure at operationSite-1 is an impact consequence of fire
Day 4, Event Cycle 9:Occurrence of contaminationOrPollutionIncident at operationSite-1 is an impact consequence of fire
Day 4, Event Cycle 9:Occurrence of noWater at operationSite-1 is an impact consequence of fire
Day 4, Event Cycle 9:Occurrence of lowPressure at operationSite-1 is an impact consequence of noWater
Day 4, Event Cycle 9:Occurrence of lowLevel at operationSite-1 is an impact consequence of pumpFailure
Day 4, Event Cycle 9:Occurrence of lowPressure at operationSite-1 is an impact consequence of lowLevel
Day 4, Event Cycle 9:Occurrence of noWater at operationSite-1 is an impact consequence of lowPressure
centralSystems is instructing inspection of operationSite-1
controller-1 is using controller-1: model-10
      conditions: [(eventOccurring lowLevel) (eventOccurring contaminationOrPollutionIncident)]
      consequences: [(eventOccurring lowPressure) (eventOccurring noWater) (eventOccurring discolouredWater) (eventOccurring
   tasteOrOdorOfWater)]
controller-1is reporting belief ['operationSite-1' (eventOccurring contaminationOrPollutionIncident)] to centralSystems
controller-1is reporting belief ['operationSite-1' (eventOccurring lowLevel)] to centralSystems
centralSystems is instructing remedial action [repairLeak] at operationSite-1
centralSystems is instructing remedial action [advisePublic takeSamples identifyContaminationSource] at operationSite-1
Day 4, Event Cycle 10:Occurrence of chlorineLeak at operationSite-1 is an secondary consequence of fire
Day 4, Event Cycle 10:Occurrence of powerSupplyFailure at operationSite-1 is an secondary consequence of fire
Day 4, Event Cycle 10:Occurrence of discolouredWater at operationSite-1 is an secondary consequence of noWater
Day 4, Event Cycle 10:Occurrence of contaminationOrPollutionIncident at operationSite-1 is an impact consequence of discolouredWat
Day 4, Event Cycle 10:Occurrence of lowLevel at operationSite-1 is an impact consequence of powerSupplyFailure
Day 4, Event Cycle 10:Occurrence of fire at operationSite-1 is an impact consequence of powerSupplyFailure
Day 4, Event Cycle 10:Occurrence of tasteOrOdorOfWater at operationSite-1 is an secondary consequence of contaminationOrPollutior
Day 4, Event Cycle 10:Occurrence of discolouredWater at operationSite-1 is an secondary consequence of contaminationOrPollutionIn
Day 4Event Cycle 9
      The events at operationSite-1 are [lowPressure fire pumpFailure lowLevel disinfectionFailure contaminationOrPollutionIncident no
      The remedial actions taken at day 4 event cycle 9 (the previous event cycle) were [advisePublic identifyContaminationSource repa
   takeSamples]
      The events eliminated were []
centralSystems is instructing inspection of operationSite-1
controller-1 is using controller-1: model-3
      conditions: [(eventOccurring fire)]
      consequences: [(eventOccurring noWater) (eventOccurring contaminationOrPollutionIncident) (eventOccurring discolouredWater)
   (eventOccurring powerSupplyFailure) (eventOccurring disinfectionFailure) (eventOccurring chlorineLeak) (eventOccurring pumpFailu
   (eventOccurring lowPressure)]
controller-1is reporting belief ['operationSite-1' (eventOccurring fire)] to centralSystems
centralSystems is instructing remedial action [extinguishFire] at operationSite-1
```

Figure 6: Excerpt from simulation transcript (spontaneous occurrence of fire at day 4)

In the simulation model reported here, correct action is action which leads to the elimination of critical incidents by remedying the events which cause and sustain those incidents. The agent's understanding is represented by that agent's models relating some events as causes to other events as effects. As is common in cognitive science, we represent the development of more sophisticated understanding as a process of *chunking* which is the combination of simple or elementary bits of knowledge into relationships and then combining those relationships into more complicated relationships. The standard reference is to Miller [14] who argued that individual humans can hold a limited number (five to nine) chunks in short-term memory at one time but that with increased understanding each chunk contains more information.

We therefore equate increased ability of the individual to understand and act within a domain of expertise with

- increased ability specifically to know when to apply relationships,
- the knowledge of a growing number of relevant relationships,
- the knowledge of increasingly complicated relationships.

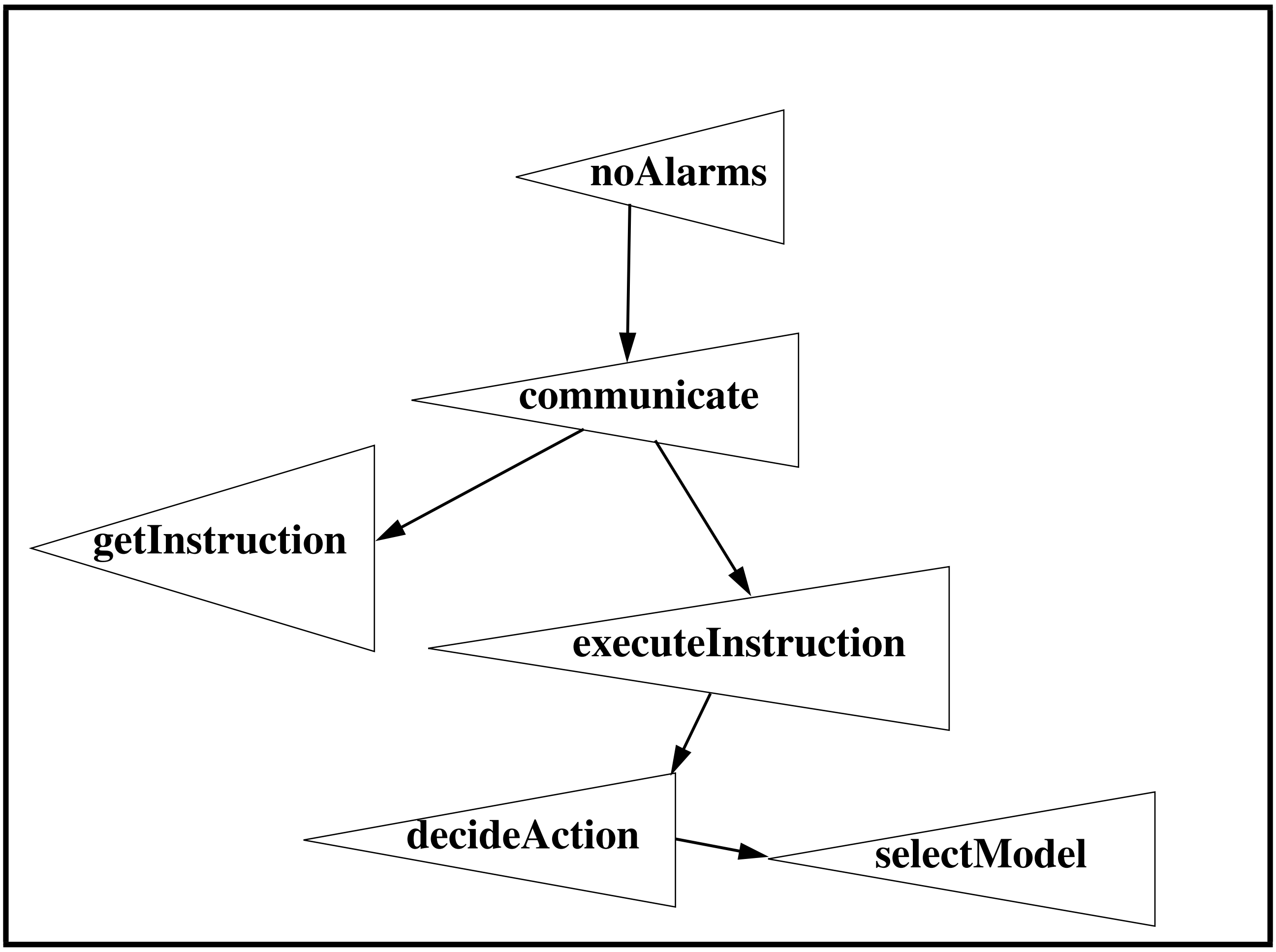

Figure 7: Problem space architecture of type Controller

The first of these criteria is represented in the simulation model by the process of model selection. This process fits into the goal structure of the agent. Agents of type Controller have the goal structure depicted in Figure 4. The adoption of the goal and each subgoal is explicitly specified by rules. So the problem space **noAlarms** is entered when there is an instruction from central systems. The problem space **communicate** is a direct consequence of the **noAlarms** problem space and **getInstruction** follows directly from **communicate**. Being in the problem space **communicate** and actually having an instruction cause the problem space **executeInstruction** to pertain. In order to execute the instruction, it is necessary to satisfy the goal **decideAction** which itself requires the goal **selectModel** to be achieved. Since the model selected determines the action to be decided, all of the procedural, cognitive work goes on in the **selectModel** problem space.

So far, all of this is consistent with the Soar and ACT-R theories of cognition. The difference comes with the representation of chunking which takes the form of model specialization and generalization as described in section 4.2. The endorsements procedure also described in section 4.2 is a rather more elaborate form of the model weighting used by Ye and Carley [26] in their Radar-Soar model.

In the **selectModel** problem space, single-cause models are candidate models if the cause is realized in the current state and the effects predicted by the model include the largest number of other currently realized events. All models specifying multiple causes are selection candidates provided that all of the causes specified by the model are currently realized and at least one of the effects of that collection of causal events is predicted by the model. The model to be used in deciding on the action to be taken is drawn from the set of candidate models with the probability proportional to its endorsement value relative to the endorsement values of the other candidate models.

The difference in results between simulations in which agents shared their understandings with one another and those in which they simply learned in parallel was not in what they learned but, rather, the efficiency of their individual cognitive behaviour.

This difference is seen in part by comparing Figure 4 with Figure 9. Figure 4 is taken from data generated in a simulation without model sharing by controllers while Figure 9 is taken from a simulation run with model sharing after the same number of simulation “days”. The models are ranked by endorsement values and the values themselves are given by the heights of the bars in each graph. The numbers are not directly comparable here because a part of the model sharing involves the sharing of endorsements, thereby to inflate the endorsement values of the shared models relative to the models which are not shared. None the less, we see the same characteristic skewness in the distribution of endorsement values of models in both cases. Where we do see a difference is in the tail of the distribution. With model sharing, the number of models used by each controller is about half the number used by agents without model sharing. In Figure 9, the smallest number of models held by an agent was 18 as compared with 33 in Figure 4. The largest numbers of models held by a single controller were 23 and 54, respectively. One consequence of this increased focus on a smaller number of successful models is in the speed of simulations: notwithstanding the additional communication among agents and more highly articulated endorsement scheme, simulations without model-sharing run at the rate of about three simulation days per hour of computer time and simulations with model-sharing run at the rate of nearly six simulation days per hour.

Although each controller informs only the controller in the succeeding shift of models which have performed exactly as specified, they do end up with much the same rank order of shared models. Moreover, as we see in Figure 10, the endorsement values of the models of each rank tend to converge. We also find that the individual cognitive activities involved in specializing models is less important than the social activity of sharing models. So much is apparent in Figure 11 where we see that for all three controllers, the most successful models, and therefore the models which determine the procedures of the organization, are shared models. A few specialized models are also shared but by and large the specialized models are the least well-endorsed. This is in part, of course, that simply because they are specialized and therefore not often used.

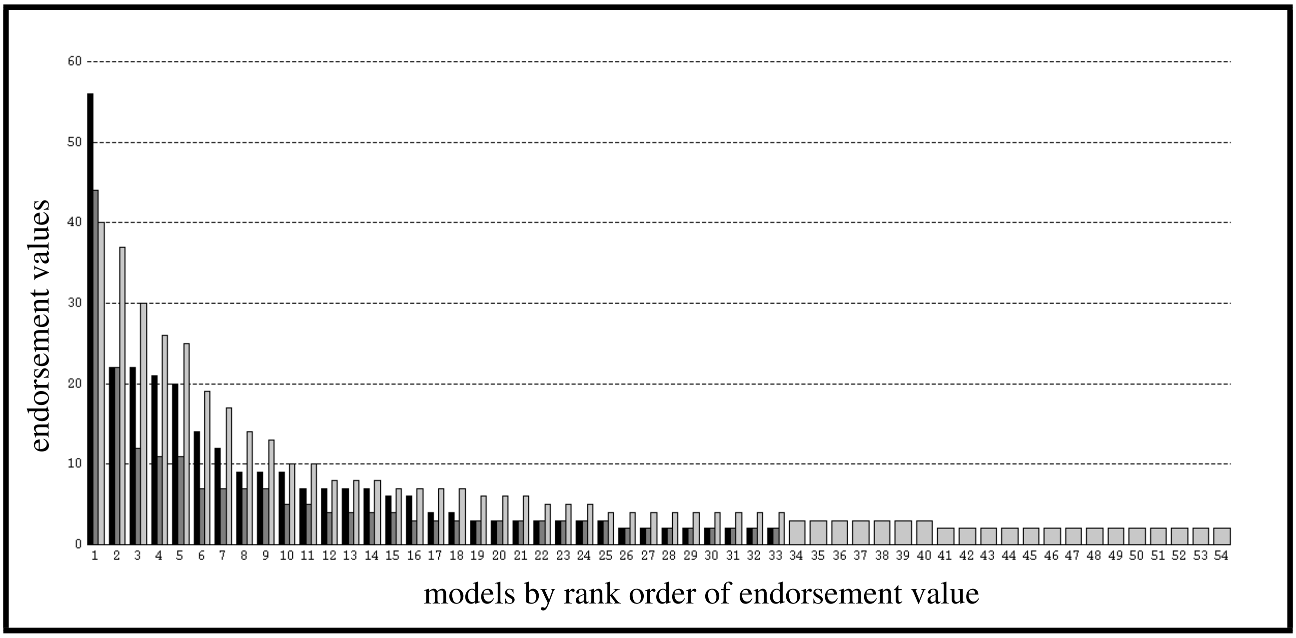

Figure 8: Model endorsement values by instances of Controller

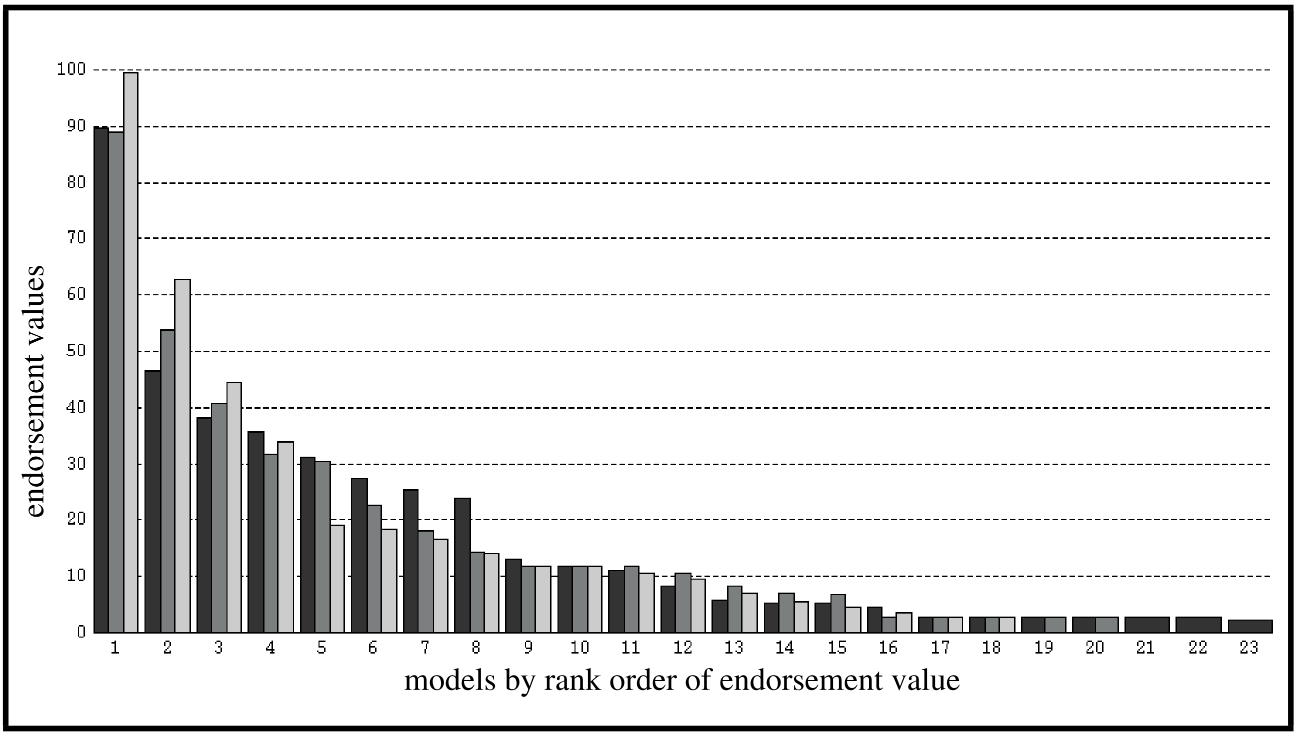

Figure 9: Model endorsement values by instances of Controller with communication

We conclude that, in the simulation experiments reported here, efficient organizational performance is enhanced by some sharing of cognitive representations of the problem space and that, for relatively simple problems, this is more important than increasing the cognitive capacities of the agents. At the same time, neither specialization resulting from the combination of predictively successful agent models nor sharing are individually or together sufficient for organization-level learning to cope more efficiently with the more complex problems.

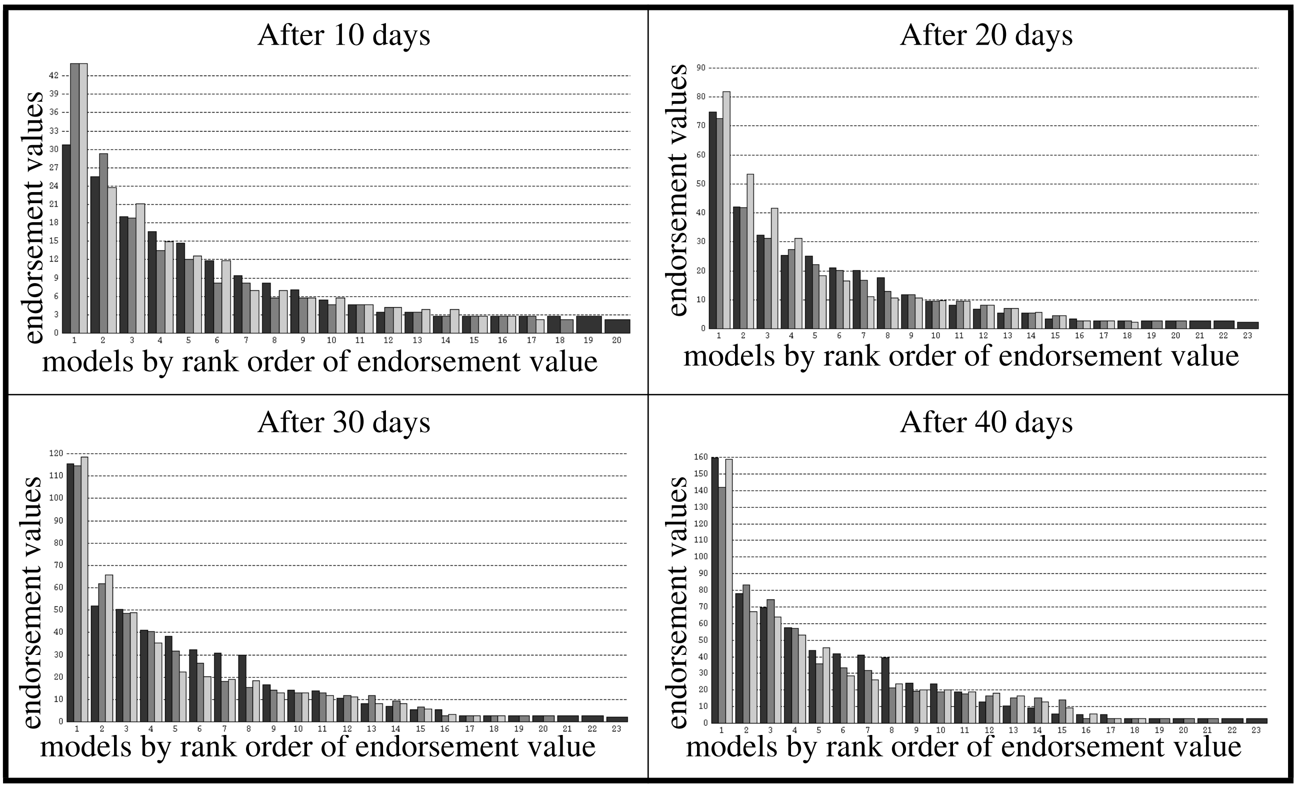

Figure 10: Developing model endorsement values over 40 days

6 Validation and verification of the North West Water model

6.1 Validation

Validation of the North West Water model is entirely straightforward. Since the model runs under SDML, it is sound and consistent relative to SDML's fragment of strongly grounded autoepistemic logic. The form of chunking used in this model has the same consequences as in Soar or ACT-R in that more complex and better performing patterns are established based on a growing history of experience. Moreover, it shares with Soar and ACT-R the problem space architecture although, because of the strictly declarative nature of SDML, the problem spaces coexist and, so, cognitive activity entails a great deal more parallellism in this model than would be possible in Soar or ACT-R.

Our finding that chunking in the form of model specialization is of much less importance than model-sharing among agents strengthens the judgement of Ye and Carley with respect to their Radar-Soar model. They turned off Soar's chunking facility because it was computationally expensive as well as of not obvious benefit. We have allowed such benefit as there is to be enjoyed but found that the extent of that benefit was far less than model-sharing. We conjecture that chunking is much less important in an organizational setting than in individual activities such as skill acquisition.

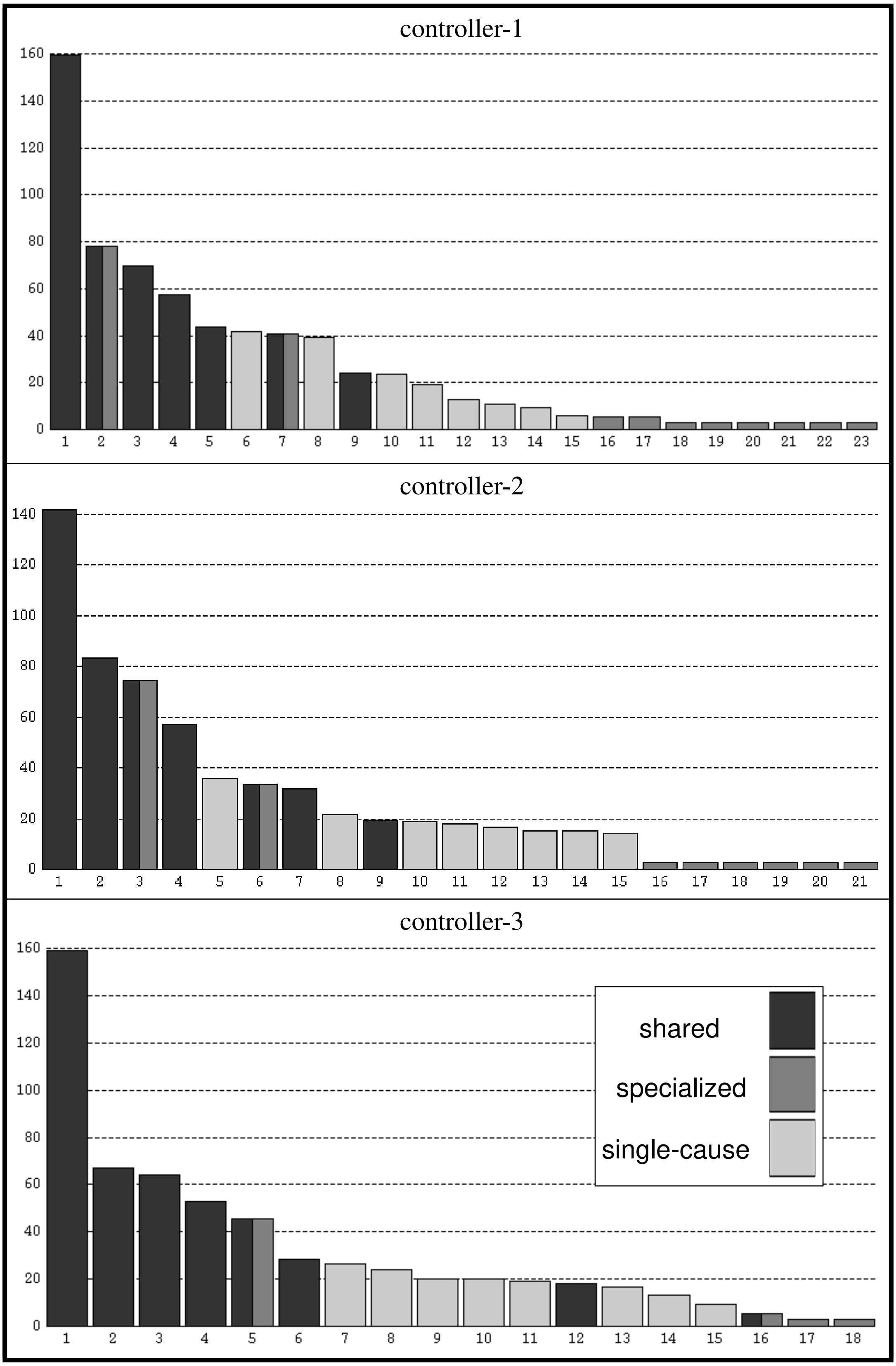

Figure 11: Endorsement values for shared, specialized, and elementary models

In relation to the management sciences in particular, the model is also well validated — though that validation does amount to a more formal encoding of some of the literature than was intended by its authors.

Consider, for example, the dominant managerial logic identified by Prahalad and Bettis [20]. This dominant logic "is a mind set or a world view or conceptualization of the business and administrative tools to accomplish goals and make decisions.... It is stored as a shared cognitive map (or set of schemas) among the dominant coalition. It is expressed as a learned, problem-solving behaviour." Although we have not been concerned here with senior management in particular, the model does give a formal (relative to FOSGAL) representation of the schema and the learned, problem-solving behaviour which define the dominant logic in the sense of Prahalad and Bettis. In these models, moreover, the representations are well grounded in cognitive science (Soar and ACT-R) The process of model development by the agents representing the network controllers of North West Water also fits neatly within Huber's [9] taxonomy of organizational learning. It amounts to knowledge acquisition through experiential learning through organizational experiments and self-appraisal.

6.2 Verification

Verification is more difficult than validation with this model because it is designed to generate a large number of critical incidents as a form of accelerated testing. We would by no means expect to see in several years the number of spontaneous and consequential critical events that this model generates at five operation sites in a simulation "day". Also, this paper does not report a full model including articulated representations of cognition in the work scheduling and planning sections, in operations control or central systems. However, the means of verifying the more complete model are straightforward. The problem space architecture for each relevant section of the organization can be obtained by interview. In many cases, particularly with planning and scheduling functions, much of the procedure is implemented as computer software and, so, is accessible for direct inclusion in a computer-based computational model. With that background, testing can take the form of simulating actual events as they have occurred in the past and about which records have been kept.

7 Summary and implications for further research

7.1 Summary

We have demonstrated in this paper that rigour and testability on the one hand and, on the other, expressiveness and richness of the language of discourse are not mutually exclusive. The use of declarative modelling techniques and programming languages supports the development of models which are consistent and sound relative to specified formal logics. Implementing such models entails both logical rigour and the greater expressiveness which is possible with logics other than mathematical formalisms or strictly numerical computational models. The virtue of declarative models with explicit correspondences to logical formalisms is that no assumptions are hidden or ambiguous in their effect. The applicability of this approach to real businesses was demonstrated by the description and results obtained with a model of an actual company. This model was also used to demonstrate that strictly declarative computational models are readily implemented to capture elements of cognitive science in a way which conforms both to the literature on organizational learning and the literature on business strategy., The particular import of the conformity of the reported model to those branches of the management science literature is that it demonstrates clearly that, although they have heretofore been developed in non-formal terms, they are naturally and rigorously implemented in computational models with explicit formal properties. We do not claim the same expressiveness of our models as are found in the non-formal literature, but we do claim to have captured *with no loss of rigour* the same relations with more expressiveness than has been found previously in formal models of organizational learning or strategy. Clearly, the models reported here are rigorous relative to different formalisms than have been found previously in the management science literature, but they are not less rigorous for that.

7.2 Further research directions

Since the purpose of this paper is to use a detailed and extended example to demonstrate that computational models can have sound formal underpinnings and support useful applications, it is appropriate to conclude by identifying useful directions for future research in both the development of the formalisms and the development of the application.

7.2.1 Formalisms

The use and indeed the development of different logical formalisms to represent agent behaviour in different circumstances or behaviour marked by different sorts of cognitive limitations has been common currency in the science of artificial intelligence since the infancy of that discipline. While the word “logic” has wide currency in the management sciences and especially the business strategy literature, models which are known to be sound and consistent relative to any logical formalisms are not to be found. For reasons described in detail in section 2.1, autoepistemic logic appears a natural logical formalism to use in modelling processes of organizational learning, development and change. The particular modelling techniques reported here also appear to be able to represent concepts which appear in the business strategy literature. We acknowledge that other logical formalisms could be more appropriate or that the some other fragment or reliance on less strongly grounded autoepistemic logic could lead to more expressive or more easily applied models. These seem to us to be an issue which needs further investigation.

We also note that a virtue of formal logics is that they support the proof of theorems. The model reported here is too rich to provide a starting point for proving theorems in FOSGAL about organizations. We nonetheless believe that such theorems should be attempted when feasible if simulations turn up apparently consistent results with different parameter settings. In general, the proof of such theorems is too difficult for all but highly trained logicians working on relatively simple cases. The use of computer-based theorem provers in conjunction with SDML or some other strictly declarative language to identify propositions which are general relative to the chosen logical formalism seems a promising avenue for the development of model validation techniques.

7.2.2 The application

It is usual in the literature on crisis management to distinguish between the crisis and the incident. An incident is a collection of events that resolved by routine procedures in a short period of time. When the number of events occurring simultaneously and the complexity defy resolution by routine means, a crisis prevails and extraordinary measures are brought into play. Like many organizations with a responsibility for public safety, North West Water has a standing crisis management team.

While we have found that routine procedures will eventually resolve the most difficult and complex episodes that arose in any simulations, in the real world it is often not sufficient to resolve them eventually. Speed of resolution is necessary to contain costs and in cases of the type considered here to prevent serious hazards to the environment and to public health. A natural extension of the computational model with the sharing of agent models is to investigate the relative computational loadings resulting from increasing the cognitive requirements on controllers and establishing a crisis management team which becomes active when certain events or combinations of events occur or when an incident goes on for some critical length of time.

The literature on crisis management as reviewed by, for example, Jacques and Gatot [10] indicates that complexity is often an aggravating factor in the development of crises. One consequence of complexity is that information and judgements are not shared or they are misinterpreted and these lacunae are swamped by the amount of information that is available. The computational models reported here already capture the essential aspects of these phenomena. To develop the models to analyse their resolution as a natural next step.

References

[1] Anderson, J.R. (1993), *Rules of the Mind* (Hillsdale NJ: Lawrence Erlbaum Associates).

[2] Armstrong, J.S. (1978) Forecasting with econometric methods: Folklore *versus* fact", Journal of Business, v. 51, pp.

[3] Bunn and Wright, 1991 "Interaction of judgmental and statistical forecasting methods: issues and analysis", *Management Science*, v.37, pp.501-518

[4] Collopy, F. and Armstrong, J.S. (1992), "Rule-based forecasting: Development and validation of an expert-systems approach to combining time-series extrapolations", *Management Science*, v..38, pp.1394-1414.

[5] Cooper, R., J. Fox, J. Farringdom and T. Shallice (1996), "A systematic methodology for cognitive modelling", *Artificial Intelligence*, v. 85, pp. 3-44.

[6] Davenport (1993), *Process Innovation - Reengineering Work Through Information Technology* (Boston, MA: Harvard Business School Press).

[7] Edmonds, B., S. Moss and S. Wallis (1996), "Logic, reasoning and a programming language for simulating economic and business processes with artificially intelligent agents" in Ein-Dor, Phillip (ed.) *Artificial Intelligence in Economics and Management* (Boston: Kluwer Academic Publishers), pp. 221-230,

[8] Hammer (1990), "Reengineering work - don't automate, obliterate", *Harvard Business Review*, v. 68, 4, 104-112

[9] Huber, G.P. 1991, "Organizational learning: The contributing processes and the literatures", *Organizational Science*, v. 2, pp.

[10] Jacques, J-M and L. Gatot (1994), "Les facteurs aggravants et les crises technologiques dans le secteur de la chimie", *Environnement et Société*, no. 12, pp. 39-56.

[11] Konolige, K. (1988), "On the relation between default and autoepistemic logic", *Artificial Intelligence*, v. 35, pp.343-382.

[12] Laird, J.E., A. Newell and P.S. Rosenbloom (1987), "Soar: An architecture for general intelligence", *Artificial Intelligence*, v. 33, pp. 1-64.

[13] Levitt, B. and J.G. March (1987), "Organizational learning", *Annual Review of Sociology*, v. 14, pp. 319-340.

[14] Miller, G.A. (1956), "The magic number seven, plus or minus two: Some limits on our capacity for processing information", *Psychological Review*, v. 63, pp. 81-97.

[15] Moss, S. , M. Artis and P. Ormerod (1994), "A smart macroeconomic forecasting system", *Journal of Forecasting*, v. 13, pp. 299-312,

[16] Moss and Edmonds (1997), "Modelling economic learning as modelling", *Cybernetics and Systems (forthcoming).*

[17] Moss, Gaylard, Wallis and Edmonds (1997), SDML: A multi-agent language for organizational modelling, *Workshop on Computational and Mathematical Organization Theory*, San Diego.

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

[19] Penrose, E. (1959), *The Theory of the Growth of the Firm* (Oxford: Basil Blackwell).

[20] Prahalad, C.K. and R.A. Bettis (1986), "The dominant logic: A new linkage between diversity and performance", *Strategic Management Journal*, vl.7, pp.485-501.
[21] Samuelson, P. (1949), *The Foundations of Economic Analysis* (Cambridge MA: Harvard University Press).
[22] Tambe, M. and P.S. Rosenbloom, (1996) "Architectures for Agents that Track Other Agents in Multi-agent Worlds", *Intelligent Agents, II, Springer Verlag Lecture Notes in Artificial Intelligence* (LNAI 1037).
[23] Turner, D.S. (1990), "The role of judgment in macroeconomic forecasting", *Journal of Forecasting*, v. 9, pp.315-345.
[24] Wallis, K. (1993), "Comparing macroeconometric models: a review article", *Economica*, v. 60, pp. 225-237.
[25] Weick,K. E. (1995) *Sensemaking in Organizations*, (London: Sage Publications).
[26] Ye, M. and K.E. Carley (1995), "Radar Soar: towards and artificial organization composed of intelligent agents", *Journal of Mathematical Sociology*, v. 20, pp. 219-246.