In G. Kadoda (Ed). Proc. PPIG 13 Pages 191-204

The Model Matters: Constructing and Reasoning with Heterarchical Structural Models

Dan Diaper
School of Design, Engineering & Computing,
Bournemouth University,
ddiaper@bournemouth.ac.uk

Keywords: POP-I.A. *team structure* POP-II.B. *problem comprehension* POP-V.A. *mental models*

Abstract

An office pool scenario is used to evaluate construction and reasoning with two types of tree diagram, a table and Simplified Set Theory for System Modelling (SST4SM). SST4SM is a formal method that has been designed for non-mathematicians. The scenario requires a non-hierarchical, i.e. heterarchical, structural model. The four approaches are ranked on various construction and reasoning tasks and the rankings' rationale is described. Overall, the tree diagrams, for all their ubiquity, are shown to be harder to construct and reason with than either the table or SST4SM.

Introduction

Structural models describe static properties. They are common as classification systems and generally provide the skeleton for more sophisticated models. The issues raised in this paper are relevant when structural modelling quality is important, e.g. in software engineering.

The word "hierarchy" is commonly used in two ways, tightly or loosely. Loosely, a hierarchy uses a concept of levels which vary in their abstractness and each level represents a potentially complete description of the world (Diaper, 1984). Tightly defined, in a hierarchy each node is the child of one higher level node and the parent of one or more lower level nodes, excluding the top and bottom levels where nodes are either a parent or a child, respectively. There are no links at the same level as a node. In this paper, "hierarchy" will reserved for tight definition models, and "heterarchy" for those that violate the tight definition.

A hierarchical model specifies that the world is organised in a particular way. The natural world, as opposed to our artefactual one, is rarely organised hierarchically. Lee and Chen (2000) provide a typical example of a heterarchical tree representation of a company's "finance inference structure" (p465) which uses additional links between levels. An argument for hierarchical models in software engineering is that they are a deliberate simplification of the real world. Unfortunately, that it is likely that such models are structurally invalid, is often forgotten.

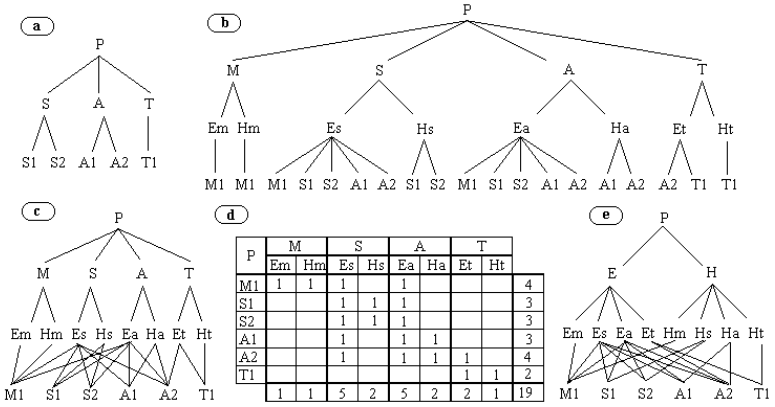
When a hierarchical model is found not to match the real world, then it might be changed by adding nodes, often to allow combinations of higher level nodes and occasionally by even radically restructuring it. The alternative is to abandon the strictly defined hierarchical structure.

A heterarchy preserves the notion of levels, only permitting relationships between higher or lower ones. Irrespective of their format, by removing some formal restrictions on tightly defined hierarchies, heterarchies are levelled models in which nodes can have multiple parentage. A tree heterarchy can model worlds that are non-hierarchical by two means by: (i) repeating nodes; or (ii) adding links.

An analyst constructing a hierarchy must: (1) have identified all the nodes; (2) located all the nodes at a level; and (3) identified the child-parent relationships between nodes. How analysts achieve these three, necessary things can vary enormously, not least because people can solve logic problems without using formal logic (Johnson-Laird, 1983).

This paper's example hierarchical system of an office pool (P) is very small and the construction and use of models will be considerably, probably exponentially, exaggerated in larger models. Initially, P has five people, two secretaries (S1 and S2), two administrators (A1 and A2) and a telephonist (T1). The minimum, hierarchical model, represented as a tree in Figure 1a, is for general management purposes. It might be used in a Business Process Reengineering exercise and as part of a software requirements analysis.

The initial pool model (Figure 1a) can be extended in three possible ways which can be related to plausible, real world scenarios and which require a heterarchical model.



P	M		S		A		T		
	Em	Hm	Es	Hs	Ea	Ha	Et	Ht	
M1	1	1	1		1				4
S1			1	1	1				3
S2			1	1	1				3
A1			1		1	1			3
A2			1		1	1	1		4
T1							1	1	2
	1	1	5	2	5	2	2	1	19

Figure 1. Models of the office pool example.

First, nodes can be added at a level. If a person (M1) is appointed as the pool's manager and who also does some of the secretarial or administrative work, which might occur if M1 has the job title of 'senior secretary' or 'senior administrator', this requires a heterarchical model.

A second way that can require heterarchical representation is to introduce a new level. To be more detailed about each role's tasks, initially one might classify tasks as Easy (E) or Hard (H). Plausibly, perhaps, all the office staff (S1, S2, A1 and A2) and the Manager (M1) can do all the easy office tasks (Es and Ea), but only those in the role (S or A) can do its hard tasks (Hs and Ha, respectively).

A third way is to add links. For example, if administrator A2 is trained to do the easy telephone tasks.

Figures 1b-d show how two tree diagrams and a table represent these extensions. Figure 1b represents nodes with multiple parentage by repeating the child nodes, whereas 1c achieves this by repeating links.

Tables like Figure 1d are logically complete as they represent every possible combination of directional link (table content) between nodes (row and column headings). Their two dimensional limitation is overcome by using sub-tables. Structuring and extracting sub-tables, even with a spread-sheet, can be complicated, and is formally that of entity relational modelling. Trees can be formally represented most easily by graph theory. While the formal approaches are powerful, they are restricted to those who are comfortable with mathematics and logic.

Simplified Set Theory for System Modelling (SST4SM)

Simplified Set Theory for Systems Modelling's (SST4SM) primary design goal (Diaper, 2000) was for a formal method for the mathematically challenged. Set theoretic equations had to be generated automatically and be understood and used without recourse to the laws of set algebra, which involves doing, in every sense, hard mathematics.

SST4SM's notation is a sub-set of set theory's, and its method replaces algebraic manipulation by the simple operations of filtering and substitution. SST4SM is based on a set model where all sets can intersect. It is therefore a flat modelling method, which represents levels by temporary structures.

SST4SM equations are clumsy compared to normal set theory usage as there is no algebraic reduction in SST4SM. To ameliorate this, the * operators indicate all possible intersecting sets in the model that are empty (∅). It is always possible to rewrite an SST4SM equation as a set theory one.

Provided sensible set names are chosen, then a computationally trivial sentence slot filling program could generate reasonably naturalistic, if a bit clumsy, English like sentences to represent its equations. This approach was successfully used with the original TAKD method (Diaper and Johnson, 1989).

The SST4SM model in Table 1, equivalent to Figures 1b-d, specifies 12 sets of interest (nodes) in expression [1], which together form the pool (P) which is the whole model of interest (U = P). Expressions [6] to [10] show the set membership of the staff and these are logically identical to the numbered cells in the tabular model (Figure 1d). To view the empty cells the * part of each expression can be written in full, e.g. [6] M1 ∈ {Em ∩ Hm ∩ Es ∩ Ea} \ {Hs ∪ Ha ∪ Et ∪ Ht}

Expression [9] shows an example of optional substitution. While SST4SM models are flat, analysts can choose to represent levels equivalent to those in the other models by using expressions [2] – [5].

The prose description of each expression is the sort of English that could be produced by a simple program that contains a small number of sentence templates and a couple of trivial grammar rules.

- *The universe of the model is P, which is made up of M and S and A ... and Ht.*
  [1] U = P = M ∪ S ∪ A ∪ T ∪ Em ∪ Hm ∪ Es ∪ Hs ∪ Ea ∪ Ha ∪ Et ∪ Ht
- *The role M is made up of Em and Hm.*
  [2] M = {Em ∩ Hm}*
- *The role S is made up of Es and Hs.*
  [3] S = {Es ∩ Hs}*
- *The role A is made up of Ea and Ha.*
  [4] A = {Ea ∩ Ha}*
- *The role T is made up of Et and Ht.*
  [5] T = {Et ∩ Ht}*
- *The person M1 can do tasks in categories Em, Hm, Es and Ea.*
  [6] M1 ∈ {Em ∩ Hm ∩ Es ∩ Ea}* = {M ∩ Es ∩ Ea}*
- *The people S1 and S2 can do tasks in categories Es, Hs and Ea.*
  [7] {S1, S2} ∈ {Es ∩ Hs ∩ Ea }* = {S ∩ Ea }*
- *The person A1 can do tasks in categories Es, Ea and Ha.*
  [8] A1 ∈ {Es ∩ Ea ∩ Ha}* = {Es ∩ H}*
- *The person A2 can do tasks in categories as those of person A1 and in category Et.*
  [9] A2 ∈ {Es ∩ Ea ∩ Ha ∩ Et }* = {A1 ∩ Et}*
- *The person T1 can do tasks in categories Et and Ht.*
  [10] T1 ∈ {Et ∩ Ht}* = T

Table 1. The SST4SM version of the extended office pool.

Constructing & Reasoning with Heterarchical Models.

Four structural heterarchy modelling techniques have been introduced that appear adequate: (1) Trees with nodal repetition; (2) Trees with link repetition; (3) Tables; and (4) SST4SM. Other formal approaches tend to be too hard for the mathematically challenged.

The four techniques are evaluated on their construction and reasoning ease, starting from a list of staff and their tasks. A simulation method was used as experiments would require that subjects have equivalent expertise. The models are to be constructed on a computer using widely available generic software, i.e. a drawing package for the trees, a spread-sheet for the table and a word processor for SST4SM.

Four construction and four reasoning tasks are set to each of the four techniques. Each of these tasks is broken into sub-tasks. The four techniques were ranked on the relative ease of each sub-task; 1 – easiest; 4 – hardest. A ranking method is easier than a cardinal scoring system and generally does not rely on making difficult decisions. The results, however, are not greatly affected by different scoring methods.

Constructing Heterarchical Structural Models

Four aspects of model construction are considered: (1) Design; (2) Construct; (3) Modify; and (4) Test. Each has a number of sub-tasks, listed in Table 2. Overall, the SST4SM model is the easiest to construct, followed by the tabular model. The tree diagrams are harder to construct; the nodal repetition tree is particularly difficult to draw.

		SST4SM	Table	Link Repetition Tree	Nodal Repetition Tree
Design	Identify nodes	2.5	2.5	2.5	2.5
	Identify levels	3	3	3	1
	Order levels	3	3	3	1
	Rank sum	8.5	8.5	8.5	4.5
Construct	Draw structure	4	3	2	1
	Order data	4	3	1.5	1.5
	Enter data	4	2	1	3
	Rank sum	12	8	4.5	5.5
	Total rank sum	**46.5**	**32.5**	**29**	**22**

		SST4SM	Table	Link Repetition Tree	Nodal Repetition Tree
Modify	Add data	4	3	1	2
	Add relationship	4	1	3	2
	Add node to a level	4	2	3	1
	Add new level	4	2	3	1
	Re-order levels	4	2	3	1
	Rank sum	20	10	13	7
Test	Detect	2	4	1	3
	Correct	4	2	2	2
	Rank sum	6	6	3	5

Table 2. Rank score data for the ease of construction of the models.

- *Design*

Starting with the data list, each method first requires some design. All the methods require the nodes (rows/columns in tables and initial sets in SST4SM) to be identified and so are ranked equally in Table 2. This is all that SST4SM requires to immediately produce its model.

The levelled models have to have their levels identified and then ordered. Ordering levels can involve a major decision, i.e. a difficult one. Figure 1e shows an alternative tree model to 1c in which the task level (Hard-Easy) has priority over the role level. Not only may such decisions be difficult, but the level order can affect how easy or difficult it is to answer questions, e.g. it is easier to answer 'How many people can do the Hard tasks?' with Figure 1e than with 1c because the left hand side of the tree can be ignored in 1e. There is no difference in assigned rank between the three levelled models.

- *Construct*

To build each model involves three sub-tasks. First, the structure needs to be created. Second, the data, i.e. the terminal leaf nodes, need ordering. Third, the data must be entered.

As a flat model, SST4SM's only structuring, which is optional, is the list of its initial sets. In the table, the column headings of levels and a generic row for the data needs to be created. With the multiple link tree it is relatively easy to draw the nodes as a legal, if inelegant, heterarchy but it would be unusual for analysts not to use an ordered list. The table and SST4SM can easily re-order their data using cut and paste which, without specialised software, is not easy with the link repeating tree. With the nodal repetition tree, however, it is basically impossible to draw the nodes without entering the data because the graphical layout is determined by the width taken up by the terminal leaf nodes.

Entering data is easiest with the table and is a common spread-sheet operation. Drawing the links with the link repetition tree is also easy as these trees cannot be graphically regular. With SST4SM, data entry basically involves writing out the SST4SM expressions that are populated by data elements. As suggested above, the nodal repetition tree is much harder to draw.

Overall, the four techniques' ranking on construction ease is: (1) table; (2) SST4SM; (3) link repetition tree; (4) nodal repetition tree.

- *Modify*

Often models are constructed piecemeal. The modify sub-task involves adding data: (i) a terminal leaf node; (ii) a link; and, finally, adding (iii) new, higher levels nodes, (iv) new levels and (v) re-ordering levels. Modifying involving deleting or moving nodes or links requires similar operations to the adding data ones.

Adding new staff data is easiest with the table by just adding another row. Next easiest, SST4SM just writes a new expression, or re-uses an existing one. The link repetition tree can add data at the ends of the terminal leaf level, but if data is inserted in the middle of the level, then up to half the links will have to be redrawn. It is much harder to add data to the nodal repetition tree as up to half of the tree must be redrawn and if it's to be symmetrical, then all of it might need redrawing.

Adding a link is easy with the link repeating tree, e.g. if the Easy administrative (Ea) tasks were to be members of both A and S roles, then a link is added in Figure 1c between S and Ea. SST4SM is perhaps fractionally less easy as it adds an additional set to an existing expression. Expression [3] can be re-written as: $S = \{Es \cap Hs \cap Ea\}\backslash^*$

The Figure 1d table has a problem with adding this link because its column headings are exploiting the hierarchical nature of the upper levels of the model. A sub-table model would be better, but the upper column headings could be changed from 'M, S, A and T' to 'M, S, S+A, A and T' with the new column heading over the Ea column. Again, the node repeating tree requires redrawing to add such a link.

As a flat model, the three types of changes to levels don't affect SST4SM. The link repeating tree is relatively easy to change with graphical cut and paste operations because whole sections will not be affected. The tabular representation is harder because the column headings require restructuring, which can be difficult. Changing the levels structure is very much harder in the node repeating case than with the other three models and usually requires extensive redrawing.

Overall, of the sort of modifications analysts might make during model construction, SST4SM is easiest. The link repetition tree and table rank similarly. The spatial constraints on the nodal repetition tree model make it difficult to modify without extensive redrawing.

- *Test*

Data entry errors are always a problem and while the four methods enter their data differently, they all appear equally prone to the problem. The example uses a missing data error, but wrongly or over assigned data give similar results across the four modelling techniques.

By error, A2's Easy secretarial (Es) tasks have not been entered. To test for this, some of the original data set's properties should be calculated. The obvious property is the number of tasks for each of the staff and their sum. With the tabular model, missing data is easy to detect as the table's grand

sum will be incorrect. Identifying the error reasonably precisely can be done using the row and column totals.

Next easiest is the nodal repetition tree, if drawn regularly. In figure 1b the links to the terminal leaf nodes all have an identical pattern depending on the number of links (1, 2 or 5). Analysts can use this property rather than counting links to higher level nodes. N.B. In 1b, to save space, this regularity has been lost with the higher links.

Checking is not difficult with SST4SM, but it requires an operation not used elsewhere. The number of expressions, i.e. on the left of the equation before the *, can be counted for each member of staff and compared with the calculations from the data list as with the table. This sort of facility would no doubt be provided by a SST4SM CASE tool.

The link repetition tree cannot be drawn regularly and there can't be an equivalent to the regular patterns of a node repeating tree. There is therefore no short cut to counting each set of links. Even totalling the lower level links is harder than in 1b, because of the crossovers in 1c.

Correction is equally easy with the link repetition tree, the table and SST4SM but, as before, is much harder with the nodal repetition tree.

The tabular format is ranked best. The node repeating tree is next best for detecting errors, but is hard to modify. SST4SM can provide a testing mechanism as easy as the table's, but requires an unusual process. SST4SM is still ranked more highly than the trees. Error detection is poor with the link repeating tree but correction is easy.

Reasoning with Structural Models

Reasoning with these models involves identifying parts of them, by induction or deduction, and combining these. While all four techniques can be used for reasoning, they are not equally easy to reason with. To demonstrate this, four different questions have been chosen that cover the majority of question types that might be posed by analysts. The first is inductive, i.e. moving from lower to higher levels and the second is deductive, i.e. *vice versa*. The third is a test of implicit pattern detection and directly arises from the second question. The fourth question is a test of the models' abilities to detect patterns on request. Table 3 shows that, overall, the table and SST4SM are equally easy to reason with compared to the two tree models, which are ranked equally.

- *Induction*

The inductive question is 'Which tasks can the staff members A1 and A2 perform?' Answering this involves: (i) identifying nodes (A1 and A2); (ii) listing the higher level tasks (Es, Ea & Ha for A1 and these plus Et for A2); and (iii) combining the task lists (Es, Ea, Ha & Et).

In Table 3, identifying nodes A1 and A2 is ranked equally easy with the table, SST4SM and the link repeating tree, but the node repeating tree has to have it's staff level exhaustively searched for the target nodes. Searching for A1 and A2 can be parallel or serial with this tree.

The SST4SM expressions [8] and [9] for A1 and A2 list the required tasks. In contrast, with the table the A1 and A2 rows would generally be extracted as a sub-table, although as the rows are adjacent in Figure 1d this isn't necessary in this case. With the link repeating tree tracing the links from each target node is not too difficult, but they will probably be written down as a list. An easier method is to mark each task node connected to the staff nodes. This is logically identical to how the tabular model solves the question. It does, however, require the analyst to recognise this easier approach. With the node repeating tree, which is terrible for identifying the target nodes, it is easier to trace its links because they don't cross, it has some symmetry and it uses much more space. The nodal repetition tree has therefore been ranked as easier than the link repeating one on listing nodes on this inductive question.

		SST4SM	Table	Link Repetition Tree	Nodal Repetition Tree
Induction	Identify nodes	4	2	2	2
	List nodes	3	4	2	1
	Combine	2.5	2.5	1.5	1.5
	Rank sum	10.5	7.5	5.5	4.5
Pattern recognition		2	3	1	4

		SST4SM	Table	Link Repetition Tree	Nodal Repetition Tree
Deduction	Identify nodes	2	2	2	4
	List nodes	2.5	4	2.5	1
	Combine	3	4	2	1
	Rank sum	7.5	10	6.5	6
Pattern Detection		4	3	2	1
	Total rank sum	**24**	**23.5**	**15**	**15.5**

Table 3. *Rank score data for the ease of reasoning with the models.*

To produce the combined answer is equally easy with SST4SM and the table. SST4SM uses a standard operation if the compressed form of [9] is not present. With the table the answer is the sub-table columns with non-zero totals. The trees act similarly at this stage but as there isn't a standard method they are ranked as harder than the other two.

Overall, on this inductive question, SST4SM ranks best, closely followed by the table. It is harder to reason with the trees and with the node repeating one in particular because of the difficulty of identifying the target nodes with it.

- *Deduction*

The example deductive question is 'Which staff members can do the easy secretarial and administrative tasks?' Answering this requires three similar sub-tasks to the inductive question: (i) identify target nodes (Es and Ea); (ii) list the lower level nodes (M1, S1, S2, A1, A2 for both Es and Ea); and (iii) combine the lower level node lists.

Identifying nodes is equally easy with both trees and the table, but harder with SST4SM. The SST4SM model must be searched for occurrences in the expressions for either Es, Ea or both. With a simple word processor the analyst would probably do two searches, although a more sophisticated search engine would allow one search. The output of a single search is the answer. With two searches then combining the two outputs is easy and a typical SST4SM operation.

On the second and third sub-tasks, the table deals with these as in the first question, substituting rows for columns. Unlike with the inductive question, the node repeating tree is well organised for answering deductive questions. The lists for Es and Ea are easy to extract by simple cut and paste from the tree. The link repeating tree is thus ranked worst at both the listing and combining sub-tasks.

The difference between the tree models is that the link repeating one is bi-directional, being equally easy for both induction and deduction. In contrast, the nodal repetition tree is uni-directional, being good for answering deductive questions but bad at answering inductive ones. The table is bi-directional, by selecting on either rows or columns.

- *Pattern Recognition*

Regularities in a model should be automatically brought to analysts' attention. In the deductive question, the staff members for the easy secretarial and easy administrative tasks are the same. On how likely the models are to alert analysts to this, the table is ranked best because its Es and Ea sub-table column and row totals will be identical. Graphically it is easy to see the corresponding cells in the sub-table.

The node repeating tree can also easily alert the analyst to this identity, if the order of the terminal leaf nodes is maintained. This tree is ranked as less easy than the table because the analyst has to look for the identity, whereas with the table it is explicitly and automatically signalled in two ways, graphically and by its row and column totals. The deductive advantages to the nodal repetition tree make it easier than the link repeating one.

SST4SM is ranked worst on this implicit pattern recognition task. In a one step search process the identity will not be detected at all unless the number of occurrences is generated. This, however, is only one alerting mechanism compared to the several generated by the table. With a two step search the analyst may be alerted in the same way as for the trees.

- *Pattern Detection*

How easy it is for analysts to explicitly search for patterns? The following question is a typical one concerning such searches, 'Which staff can do the same tasks?'

For the levelled models this is an inductive question and is therefore difficult to solve with the nodal repetition tree. In contrast, the answer to this question is explicitly present in the SST4SM model's expression [7]. The SST4SM analyst simply has to note which expressions have more than one element on the far left of the expressions.

Both tree models effectively produce a tabular model to answer this question. They are therefore worse than the tabular model itself. The brute force approach with a table is to compare pairs of rows in turn. A short cut is only to compare rows with the same total. In 1d this reduces the problem to two sets of comparisons: S1, S2 and A1; and M1 and A2. Even with this short cut, the necessary operations are obviously more complicated than the simple search necessary with SST4SM.

Discussion

It does matter what modelling technique is chosen to represent a heterarchical structural model. Heterarchical models are quite common and are less well understood than hierarchical ones. Structural models are ubiquitous, alone or as the skeletons for more sophisticated types of model. Formal methods such as graph and set theory are available, but their use is limited to those with adequate mathematical skills. SST4SM is a formal method for non-mathematicians.

Examining the minimum operations of an efficient analyst to construct and reason with a heterarchical model, represented as a node or link repeating tree, as a table and by SST4SM, shows a clear advantage to the non-tree models. Nodal repetition trees are much harder to draw than link repeating ones. The latter are not easy to reason with because they cannot easily represent many regularities that may be in data. Link repetition trees, however, can equally answer a broader range of questions than the node repeating ones, which are better at answering deductive questions than inductive ones.

Unlike the other models, SST4SM structural models are flat. Levelled structures can be temporarily represented so that SST4SM can respond flexibly to a wide range of questions.

Tables primarily accrue their power as reasoning tools because row and column totalling is a natural and easy operation on them. Sub-table extraction can be complicated however, and may require a relational model.

The pool model used in this paper is very small and the important, earlier caveat is that "the construction and use of models will be considerably, probably exponentially, exaggerated in larger models." SST4SM is not affected greatly by model scale as its initial output table is in any case vast and enlarging the model simply causes its filtering rules to become more effective (Diaper, 2000). At the other extreme, node repeating trees, in whatever graphical style, will always be space expensive, and hence size limited, compared to link repeating trees. The limitation on the size of link repeating trees is becoming apparent, however, even in Figures 1c and 1e and clearly another couple of heterarchical levels, or just many more staff nodes, would make working with such models, particularly tracing and counting the links, a difficult, slow and probably error prone process. There is no reason why tables should be subjected to any scale limitations since the structure of all relational databases is tabular. While not as size limited as the trees, a spread-sheet of even modest size can become difficult to work with.

This paper is directed to analysts who are sufficiently concerned with their models that, at least, they will prepare them using widely available, generic software. Software engineering undoubtedly provides one example where this is, at least, usually the case. Certainly heterarchies represented as trees are not uncommon. Once they get used to it, analysts should find that using tabular models are easier to construct, edit and reason with.

SST4SM is a recent development and fares surprisingly well. An SST4SM model is itself a table, but has advantages over other tabular forms because it has been designed to model complex relationships between things. What should make SST4SM preferable to more traditional tabular formats is its formal but non-mathematical methods provide a means by which systems analysts can iteratively build their systems models (Diaper, 2000). Overall, the place from tree models is on white-boards.

The work reported in this paper may also assist those designing CASE tools. They might, for example, test their tools against the types of scenario task reported in this paper. SST4SM is still under development and the work has highlighted the requirement for it to test for data entry errors. More critically, SST4SM needs to provide a more bi-directional logic so that induction and deduction are equally easy.

References

Diaper, D. (1984) An Approach to IKBS Development Based on a Review of 'Conceptual Structures: Information Processing in Mind and Machine' by J. F. Sowa. *Behaviour and Information Technology*, 3, 3, 249-255.

Diaper, D. and Johnson, P. (1989) Task Analysis for Knowledge Descriptions: Theory and Application in Training. in Long, J. and Whitefield. A., (eds.) *Cognitive Ergonomics and Human-Computer Interaction*. 191-224. Cambridge University Press.

Diaper, D. (2000) Hardening Soft Systems Methodology. in McDonald, S., Waern, Y. and Cockton, G.(Eds.) *People and Computers XIV*. 183-204. Springer.

Johnson-Laird, P.N. (1983) Mental Models. Cambridge University Press.

Lee, C. and Chen Y.-T. (2000) Distributed Visual Reasoning for Intelligent Information Retrieval on the Web. *Interacting with Computers*, 12, 5, 445-468.