	CPML	Pseudo programming languages	PBG/ production rules
Executable	no	nearly	possible
Control structures	yes	yes	possible
Deterministic about control	optional	yes	optional
Including assumptions about mechanism	hard	very hard	easy

Literature

Our language CPML is based on the KADS modelling language (KCML). CPML is somewhat simplified (for more details, consult the references at the end of this section). The most important difference between KCML and CPML is that CPML has no restricted set of knowledge sources. The original KCML has a set of predefined knowledge sources that describe types of problem-solving steps. Research in the context of this modelling language has produced a library of models defined in terms of these knowledge sources that can be used as the basis for constructing new models. Another difference is that we have omitted a fourth layer of the KCML model, the *strategic layer*. The best references on the KADS methodology can be found in the book by Schreiber et al. (1993) and in several articles. More specifically Wielinga et al. (1992) give an overview of the KADS methodology, van Harmelen & Balder (1992) describe the logic-based formal specification language $(ML)^2$ and Breuker & Wielinga (1987) discuss the use of think aloud protocols in the KADS methodology. Several systems have been built by designing and implementing a new language. For example, the PDP system (Jansweijer *et al.*, 1987; Jansweijer, 1988) which combines concept hierarchies and goal oriented production rules and was written on top of Prolog. Brownstone et al. (1985) discuss production rule systems. Another implemented system based on production rules is SOAR (Laird et al. 1987). Newell & Simon (1972) provide an excellent and extensive discussion with examples of problem behaviour graphs and production rule models. Lucas & van der Gaag (1991) give a detailed technical discussion of production rule systems. Another psychologically motivated language based on production rules is the GRAPES system by Anderson and his colleagues (1989). A collection of special purpose problem-solving mechanisms developed for knowledge acquisition is Marcus (1988). Musen (1989a, 1989b) describes a more general method for designing special purpose problem solvers along with support tools.

Chapter 7

Analysing the protocols

7.1 Introduction

Thus far we have discussed how think aloud protocols are collected and how psychological models are constructed. We now turn our attention to the relation between the protocols and the psychological model. There are three issues that deserve attention in the analysis of think aloud protocols:

1. Constructing a mapping between protocols and model.
2. Avoiding bias and interpretation errors in comparing protocols and model.
3. Quantifying the correspondence between protocols and model.

This chapter will give standard procedures and techniques for the analysis process. Figure 7.1 lists the steps of this phase of protocol analysis. The goal is to construct a mapping between the psychological model and how the cognitive process will appear in the protocols. This mapping will take the form of a coding scheme that is based on the psychological model and a *verbalization theory*: a theory about the verbalization process. Using this, the protocol can be compared with the model. We shall describe how to construct a coding scheme and how to apply this in the context of psychological research. Finally we go into the comparison of the coded protocol with the constructed model. This chapter will be concluded by making suggestions on how to report the results of studies using protocol analysis.

In the context of knowledge acquisition objective measurement of the correspondence between protocol and model is less important than in empirical research. However, also the knowledge engineer has to specify the relation

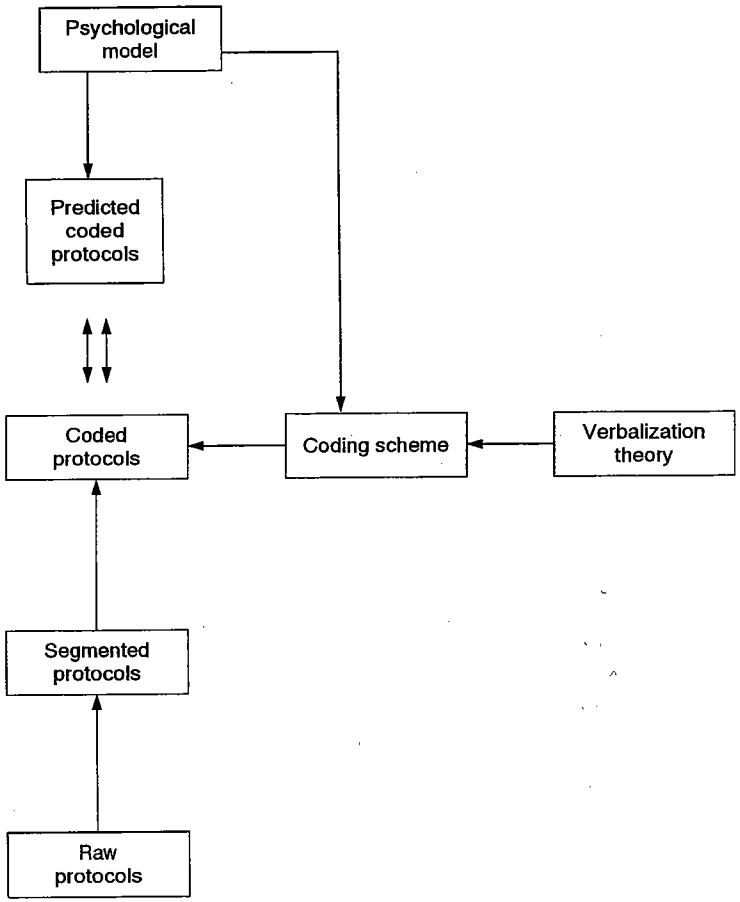

FIGURE 7.1: The analysis process

between protocols and model but it is not necessary to take measures to avoid bias in the interpretation or to quantify the correspondence between protocol and model. Therefore Sections 7.5 to 7.7 (Coding procedures, Intercoder reliability and Comparing the coded protocols with the models) and 7.9 (Reporting the results of protocol analysis) are less relevant for the reader who is only interested in knowledge acquisition. Both in knowledge acquisition and in scientific research a mapping between protocols and model is constructed. Therefore the rest of this chapter is relevant for both applications.

7.2 The role of protocols as data in research

In scientific research a distinction can be made between 'raw data', 'data' and 'theory'. Raw data are obtained by measuring procedures that are objective in the sense that they can be applied under any condition and will produce the same results. These results are generally considered as valid by the scientific community. Examples of such standard procedures are the use of certain thermometers to measure the temperature, the use of electronic clocks and hand-switches to measure reaction times and the use of certain intelligence tests and procedures for administering these. In some cases these objective data are interpreted, abstracted or aggregated in a way that is not as objective as the previous procedures. For example, the scores on an intelligence test may be combined to measure a new type of intelligence or time measurements are classified as indicators of a cognitive style by human judgment. In that case the result is less objective because it may depend on human judgment or because its validity is not generally accepted.

In the context of think aloud protocols we give standardized procedures for *segmenting* the written protocols and we argue that segmented protocols can be treated as *raw data*. The next step from raw data to the model is to *code* the segments in terms of the model. This usually requires a *coding scheme*, an extension to the model that describes how categories of the model will appear in the protocol. The *coded protocols* are *data* in the sense that coding involves procedures and coding schemes that are less objective than those for segmenting. The *coded protocol* that can be compared with the model which is directly derived from the theory.

7.3 Transcription and segmentation

It is very hard to analyse a think aloud protocol directly from audio recording. This is especially so in exploratory use of protocols. It is simply more difficult

to get an overview over audio recordings and it is also more difficult to retrieve fragments from an audio recording. Protocols are generally transcribed into text. Although we may hope that in the future this process can be automated it is now simply much work. Transcribing is easier when using a special cassette player with a foot-switch and much depends on the quality of the recording. Notes and other observations are inserted in the transcription as much as possible.

Next the protocol is *segmented*, that is divided into *segments*. Research on language production and language understanding shows that in speech the boundaries of phrases are usually marked by pauses (Ericsson & Simon, 1993). The combination of these pauses and the linguistic structure provide a natural and general method to segment a think aloud protocol. Experience with segmentation has shown that there generally exists a high level of agreement between people asked to segment a written protocol while listening to it. Segmentation becomes more difficult and less reliable when it is done on the basis of the written text only. It is a good idea to design a form for the protocol segments and the analysis and to number the segments on this form. There are many computer systems that can be used to support storing and retrieving protocol fragments. Segments can for example become items in a database and be indexed by information about the protocol (subject, problem, date, etc.) and by model categories. In the analysis, segments are often combined into *episodes*, see Section 7.5.2. An episode is a sequence of segments that corresponds to a single element in the model.

7.4 Coding scheme and verbalization theory

7.4.1 Introduction

A procedural psychological model describes *which* cognitive processes will occur and also *in which order* they will occur. There is often still a substantial gap between the model and the protocol data. This makes it necessary to extend the model with an *operational definition* of categories in the model in the form of a coding scheme. The coding scheme specifies how elements of the model can be identified in the data. In the context of knowledge acquisition it is also relevant to consider this question because the verbalization theory will tell the knowledge engineer which information can be obtained directly from the protocols, which cannot be obtained at all and which can be obtained partially or indirectly. In the context of knowledge engineering a coding scheme is not necessary to achieve objective data analysis but is used for retrieval of relevant protocol fragments. In practice one usually associates protocol fragments with model components without a mediating coding scheme.

7.4.2 Constructing a coding scheme

Because it is difficult and therefore unreliable to compare the protocol and the psychological model directly, it is usually necessary to make a coding scheme to help in the analysis. A coding scheme is based on the psychological model and and on the verbalization theory. The step from psychological model to coding scheme is usually quite straightforward. Take every (sub-)process distinguished in the model and state how you expect these processes to appear in the protocols. Take for example the word problem solving processes. The model for using the approximation method has a sub-process called 'guessing'. One could assign the coding category 'guessing' to this sub-process, and one would expect to find statements in the protocol indicating a numerical solution with a certain uncertainty. The subject will for example say: 'Maybe it is X (X = number)', 'Could it be X?' or 'Let's try X'. This would appear in a coding scheme as:

Cognitive process	Description
Guessing	'Maybe it is X (where X is a number)', or 'Could it be X?' or 'Let's try X'.

For every process described in the model one defines the *type of statement* referring to that process. Categories in the coding scheme can be described in general terms but it is usually very helpful to give some examples of prototypical statements for each category. If two or more categories are similar it helps to emphasize the difference. It is of course possible that it appears to be unfeasible to define a reliable coding scheme. In that case one can has to drop the distinction, to revise the conditions under which the data were collected or even to find a different method for collecting data.

This type of analysis of verbal reports is similar to *content analysis*. Content analysis is usually applied to written documents that are stated in grammatically correct language, which simplifies the analysis. However, content analysis has also been applied to, for example, interview texts, which brings it quite close to the analysis of think aloud protocols. The main difference between content analysis in general and the analysis of think aloud protocols is that the latter usually involve problem-solving processes and therefore involve process models. Content analysis usually concerns static properties of texts.

7.4.3 Grain size and aggregation

If the task analysis and the psychological model are very detailed then elements of the psychological model may correspond directly to segments in the protocol. However, often the model is more coarse-grained and a single element (for example a single knowledge source or production rule) corresponds to a sequence of several segments. This has no consequences for the coding scheme, because one simply defines categories that cover more than one segment.

7.4.4 Special coding categories

There are some categories in the coding scheme which are not directly derived from the model. These are the verbalizations which are not covered by the model, but may still be anticipated in the protocols. For example:

(a) Talking about not-task related issues ('Oh, I must not forget to call my friend').
(b) Evaluation of the task or task-situation at a meta-level ('It is tiring to talk so much', 'I hate these kinds of problems').
(c) Comments on oneself ('I am thirsty', 'I am not comfortable').
(d) Silent periods. At times people will briefly stop verbalizing. After some time they may continue or they may be prompted to continue. It may be relevant to assign a code to relatively long pauses.
(e) Actions. The subject performs an action (for example, writes a note or manipulates a device). It is usually best to include this in the coding scheme.

In some cases one would ignore all these events as irrelevant - because they do not bear upon task performance - so putting them in one category: 'irrelevant comments'. At other times, however, these remarks might be an indication of the level of difficulty of a sub-task or of the cognitive load of the subject. For example, if a subject makes a lot of task-irrelevant remarks each time he must do a calculation, one might suspect calculating is difficult for this person. Sometimes the content of these interruptions in task performance is not relevant, but the moment at which they occur is. For example, it may indicate that the person who solves the problem does not make progress (reached an 'impasse'). In that case, special codes should be used for interruptions.

7.4.5 Coding form

To facilitate the coding process, design a coding form that consists of the segmented protocol (with numbers for the segments) and has space for marking the code assigned to each segment and for indicating discrepancies between model and coded protocol. If the protocols are stored in a computer database this information should be added for each segment or episode.

7.4.6 Example: a coding scheme for architectural design

Let us take a look at a part of the coding scheme for architectural design constructed by Hamel (1990). This coding scheme corresponds directly to the model in Figure 5.2. For every category in this model several processes are distinguished. Take, for instance, an architect whose cognitive process at a certain moment is hypothesized to be taking place on the 'the styling schema level, execution phase'. Hamel states that this process should be reflected in the protocol as comments on the strategy and the way of working during the synthesis of the design. Such comments are, for example, verbalized in the protocol as: 'Well, I am now going to look at the consequences of this location for the playing possibilities of the kids and their safety.' Below we give examples of a few coding categories. The complete coding scheme is reproduced in Appendix E.

Code	Description
SO2	Synthesis, orientation, estimation of combining aspects of the design
AO2-10	Analysis, orientation, using one's own knowledge with regard to functions of the assignment
SU9	Synthesis, execution, isometric perspective
SE1	Synthesis, evaluation, comparison with expectations, inspection, or checking of data or requirements

These coding categories clearly require knowledge about the task. Let us now take a look at a part of the protocol fragment given in Chapter 1 to see how this was coded.

Code	Line	Protocol text
SO2	12:	`but maybe we can with er do something with that shack`
AO2-10	13:	`water I'll just put tap [notes 'tap']`

```
AO2-10  14:  what children of course
AO2-10  15:  what what what is much handier
SU9     16:  [sketches water tub]
SU9     17:  maybe something er where water comes out of er
SU9     18:  and that you turn off in winter
SU9     19:  but then it does not trouble you
SU9     20:  and then it may run into here somewhere
SE1     21:  then they can still mess about ...  with water
SE1     22:  then they can play with this
SE1     23:  that just comes it slowly drips out of it or so
SE1     24:  then they'll get dripping wet in summer
SE1     25:  and then they can get around this
```

Note that here each segment is coded. An alternative way to code this part of the protocol is to divide the protocol into episodes and assign a code to each episode. In that case, line 13-15 would be coded as `AO2-10`, line 16-20 would be coded as `SU9` and line 21-25 as `SE1`.

7.4.7 Verbalization theory

A coding scheme is based on the psychological model and on our knowledge of the way in which cognitive processes will be verbalized. In Chapter 2 we discussed the factors that influence verbalization. In particular, information that resides in working memory for a very short time, that is difficult to verbalize because of complexity or because of its non-verbal character, may not appear in a protocol. For example, a chess master analysing a chess position may well perceive a large part of the chess board as a whole. Verbalizing this requires her to construct a short linear representation that can be uttered as spoken language. This task is very hard and likely to give synchronization problems between the speed of the thought process and verbalization.

If verbalization is difficult, then verbalizations will be idiosyncratic: there will be individual differences in verbalization, even if the content of the cognitive process would be the same. Take for example tasting wine. Unexperienced wine-tasters when asked to compare different wines will talk of the flavour of the wines in their own terms, calling a taste sour or bitter. Expert wine tasters share a common vocabulary to express how wine tastes, calling the wine fruity. Idiosyncratic expressions cannot be covered by a *verbalization theory* and a coding scheme. In some cases it is possible to use part of the protocol to construct a 'personal coding scheme' for each subject. This is a good way to handle systematic differences in vocabulary between subjects. If a task involves information that is hard to verbalize then the analysis should allow incomplete verbalization and use a more abstract (or more branched) coding scheme.

7.4.8 Example of a verbalization theory

Consider the models of solving arithmetic word problems discussed in Chapter 5 and 6. Suppose that the persons solving these problems are reasonably experienced in this kind of task. What can we expect to find in the protocols? According to the model of Chapter 2, people verbalize the new contents of working memory or rather part of that. From the model we can find which information will appear in working memory. In the case of production rule models this is simply the information added by the rules and for the other languages the new information that is constructed during problem-solving is likely to appear there. For example, for the production rule model sketched in Section 6.4.3 we would find the following elements that can appear in working memory:

- The problem text (when it is read by the subject).
- Names of schemata (for example *change* and *compare*).
- Parts of schemata (for example in the *change-schema* we have: *amount-before*, *amount-after* and *amount-change*. These are either filled from the problem text or by computation).

For some expressions it is not clear if they will appear in the protocol. For example, according to the production rule model the name of a schema will be added to working memory. However, it is quite possible that schema recognition takes place implicitly and that the result does not appear in the protocols. This is plausible if there is no standard word for the *change-schema* and on the other hand this refers to the rather common concept of losing or obtaining something. This means that it is likely that people will not report this thought: it will pass very quickly and is relatively difficult to verbalize. From such considerations we can drop part of the list of possible verbalizations. This can be indicated in the coding scheme ('will/may not appear in protocol'). For the same reason the names of parts of schemata are unlikely to appear in a protocol. We will expect only the amounts.

Note that the production rules themselves do not appear in working memory. They are retrieved and applied by the cognitive machinery of the subject and do not appear as contents in working memory.

How sophisticated and elaborate should the verbalization theory be? In most cases there is simply not enough psychological knowledge and knowledge about the people involved in the task to predict what will appear in the protocol. This means that one has to resort to pilot protocols. These are used to perform the exercise described above.

7.4.9 Methodological requirements for the coding scheme

The main requirement for a coding scheme is that it allows objective coding of protocol fragments in terms of the psychological model. This breaks down into the following specific requirements:

Completeness (with respect to the model): The coding scheme must contain descriptions of all reasoning steps (or their results) that appear in the model and that can be expected to appear in the protocols on the basis of the verbalization theory. We should of course not require the coding scheme to cover all of the protocols. This coverage is the hypothesis that is to be validated. Segments (or episodes) that cannot be coded correspond to cognitive sub-processes that are not explained by the model!
Justified: The coding scheme must be justified by the model and the verbalization theory. One should not introduce new elements or concepts in the coding scheme that do not follow directly from the psychological model and the verbalization theory.
Grain size: Either the grain size must correspond to that of segments or it must be possible to objectively aggregate episodes that correspond to the categories in the coding scheme (else aggregation becomes part of coding which complicates the analysis). In the latter case aggregation should be done in a separate pass through the data. See also Section 7.5.2.
Unambiguous: The coding scheme must be clear enough to be used by outsiders. This is necessary to maintain objectivity of the coding procedure. Different coders must assign corresponding codes. We give a measure for this below.
Context independent: If a coding category describes a single cognitive process then it must be possible to recognize this without the context in which it appears. This is particularly important if we want to test the sequence predicted by the model. If the context in which a segment or episode appears is necessary to assign a code to it, it is no longer possible to really test the order because the order was used to assign the codes!

As with all methodological requirements, it is usually not possible to meet them for 100 per cent nor is it possible to construct an adequate coding scheme

in one pass. The coding scheme (and the coding procedure) must be tested and evaluated on pilot protocols before it is applied to the actual data. For some criteria it is possible to quantify the extent to which they are met.

7.5 Coding procedures

7.5.1 Introduction

Coding means assigning labels to protocol episodes following the coding scheme. The result of applying the coding scheme to a protocol (raw data) will be a *coded protocol.* Segments or episodes that cannot be coded and sequences that are not predicted by the model but that do appear in the protocol reflect deviations of the model.

7.5.2 Aggregation

Transcription and segmentation are standard procedures that can be performed without any knowledge about the task or the model. This is not true of *aggregation.* Aggregation means collecting segments into groups that correspond in 'grain size' to the model. This is necessary if the model cannot describe protocols at the grain size of segments. For example, a model may contain a component 'compute the average' that is not elaborated further. In the protocol several segments together may correspond to this process. In this case only an episode, a sequence of segments can be recognized. Strictly speaking, aggregation of segments into episodes is part of the coding process because it requires knowledge about the model and cannot be performed in a model-independent way.

7.5.3 Coding

If possible, it is best to leave the coding of the protocols to independent coders. The researcher who has constructed the model and the coding scheme usually is too much attached to a certain research hypothesis to do the coding with an objective mind. Try to find coders who are not involved in the research project, and who have no specific interest in the outcome of the protocol analysis. This gives the best guarantee for objective (reproducible) coding. Give the coders only the minimal information about the purpose of the study and instruct them to be as precise as possible. Coders need to be trained in the use of the

coding scheme. The protocols used for revising the coding scheme might be used for this.

The interpretation of a single phrase may be influenced by the context in which it appears. Take for example a subject who has been making a lot of errors and was very confused about the right way of solving the assigned word problem. If this subject would then say: 'the answer is 15', one would be inclined to interpret this step in the problem-solving as another wild guess. If the subject was someone who methodically followed a straightforward procedure, one would think it was just the outcome of the problem-solving process and thus the phrase 'the answer is 15' would be coded accordingly.

Measures one could take are: give coders only the minimal context information for coding. To refrain coders from using the context in which a protocol episode appears which might bias their coding - minimize the available context. The context involves:

(a) The subject who produced the protocol: shuffle protocols of different subjects and distribute them at random over coders.
(b) The content of the protocol: if possible cut the protocol in pieces that are just big enough to be coded reliably and shuffle these pieces. This will minimize the bias due to the context in which the fragment appears in the protocol. This is not always possible, sometimes protocol statements get unintelligible when lifted out of their contexts. In that case one must accept context-dependent coding.

7.5.4 Rating protocols or protocol fragments

If the model and the theory about the process are not procedural but structural, describing properties of protocols or protocol fragments, then protocols are rated. In this case scales are defined describing properties of the protocol and these are applied by coders following a procedure that is analogous to that for procedural models. In this case, the rating procedure produces standard numerical data and standard procedures for analysis can be applied.

7.6 Intercoder reliability

Before it is applied, a coding scheme and the coding procedure must be evaluated. In particular, correspondence between codes assigned by different coders to the same data must be found. If this correspondence, intercoder reliability, is low, then this means that the coding scheme is ambiguous. How can this

correspondence be quantified? The techniques for quantifying correspondence all use one set of data that is coded by two (or sometimes more) coders. In the case of think aloud protocols, the data consist of an entire protocol, several protocols or one or more fragments. Intercoder reliability may vary over protocols and fragments so it is important to use a representative sample of the total set of protocols.

Reliability is usually quantified over segmented protocols. The segmentation itself is usually rather reliable and coding reliability can simply not be quantified over protocols that are segmented in different ways. It is also better to exclude irrelevant parts of the protocol (for example comments on the situation, reading fragments of text) because these inflate the reliability with respect to the actual cognitive process.

The selected data are then coded following the coding scheme that was designed in advance. Usually this means that they are assigned to a coding category. From these codings a cross-table can be constructed. Each cell contains the number of elements (for example fragments) as coded by both coders.

Take, for example, a small protocol which consists of eight segments. Two coders have coded this protocol. The coding scheme has only two categories: A and B. The two coders have coded the protocol as follows:

Segment	Code by coder 1	Code by coder 2	Correspondence
line 1	A	A	yes
line 2	A	B	no
line 3	B	B	yes
line 4	A	A	yes
line 5	A	A	yes
line 6	A	B	no
line 7	A	A	yes
line 8	A	A	yes

In this example both coders coded a line 5 times as A (line 1, 4, 5, 7 and 8), 1 time as B (line 3), and 2 times coder 2 assigned a B where coder 1 coded an A (line 2 and 6):

C1	C2	Frequency
A	A	5
A	B	2
B	A	0
B	B	1

From this coding the following cross-table is constructed:

Code	A	B	Total
A	5	2	7
B	0	1	1
Total	5	3	8

Correspondence between coders is now quantified as the association between their codings. The first obvious measure is the proportion of corresponding codes with respect to all codes. For the above table that would be the sum of the number of codes on the diagonal, namely 6 (5 + 1) divided by the total number, namely 8. This gives 75 per cent correspondence.

A conceptual problem is, though, that this measure is sensitive to differences in marginal frequencies, the proportion of the different coding categories used by the coders. In this example, code A is much more often used than code B, namely seven times by coder 1 and five times by coder 2, where B is used one time by coder 1 and three times by coder 2. This means that the expectation for an arbitrary segment to be coded as A, is higher than the expectation for it to be coded as B. In this small example the need for correction for marginal frequencies might not be so obvious. But take for example a protocol of 100 segments. If 99 segments are coded as A by both coders and only 1 segment is coded as B by one of the coders, the correspondence would be 99 per cent. However, it is not fair to say that in this case an extremely high intercoder reliability has been reached. For a segment to be coded as A, the chance was 99 per cent. So in some sense the coders did not do better than if the coding had been done by a random generator. The one segment about which there was no agreement between coders is far more significant than all the other segments which were coded the same.

In another example, where from the 100 segments 50 were coded as A by both coders, and 49 as B, a reliability of 99 per cent indicates much more correspondence between the coders. In this case every segment had a chance of some 50 per cent to be coded as A. In this case the fact that the coders coded nearly all segments the same, cannot be ascribed to chance.

Several measures have been invented that to some extent correct for differences in marginal frequencies. These measures show subtle differences in behaviour and in the underlying notion of association. One frequently used measure, that we recommend, is Kappa. This measure is based on a correction for marginal frequencies and defines association as the relative proportion of corresponding codes with the following correction:

$$\text{Kappa} = \frac{\text{(proportion corresponding}-\text{expected proportion corresponding)}}{\text{(1}-\text{expected proportion corresponding)}}$$

Here the expected proportion corresponding is calculated by multiplying and adding marginal frequencies. In the example the correspondence for A that can be expected from the marginal frequencies of B is $7/8 \times 5/8 = 0.55$. For B we will get $1/8 \times 3/8 = 0.05$ and then the total is $0.55 + 0.05 = 0.60$. For the table above Kappa is:

$$\text{Kappa} = \frac{(0.75 - 0.60)}{(1 - 0.60)} = \frac{0.15}{0.40} = 0.38$$

So, in this example we have gone from a proportion of corresponding codings of 0.75, to a Kappa of 0.38. The fact that Kappa is lower is due to the fact that code A is more frequently used than B. That the difference is so great is due to the fact that this example is so very small.

What will happen to the two extreme examples of the 100 segments protocol if we calculate the Kappa? In the case of 99 segments coded as A the Kappa is:

$$\text{Kappa} = \frac{(0.99 - 0.99)}{(1 - 0.99)} = 0.00$$

In the example of 50 segments coded as A and 49 segments coded as B by both coders the Kappa is:

$$\text{Kappa} = \frac{(0.99 - 0.499)}{(1 - 0.499)} = 0.98$$

The Kappa makes a correction for the correspondence that can be expected from the marginal frequencies. This is a conservative estimate of intercoder reliability because similar marginal frequencies will make Kappa low where one could argue that similar marginal frequencies themselves indicate intercoder reliability. Since the proportion correspondence is an optimistic estimate it is best to report both proportion correspondence and Kappa.

Let us now look at a larger example. A set of 98 segments is coded by two coders: Coder 1 and Coder 2. The coding scheme has 5 categories and the table below shows how many fragments were scored as A by both, as A by Coder 1 and B by Coder 2, etc.

Code	A	B	C	D	E	Total
A	6	0	2	0	0	8
B	1	48	11	0	0	60
C	1	2	17	0	0	20
D	0	0	0	5	3	8
E	0	0	0	0	2	2
Total	8	50	30	5	5	98

The proportion corresponding codes with respect to all codes is calculated by dividing the sum of the number of codes on the diagonal (81) by the total number (98). This gives 83 per cent correspondence.

Now we calculate Kappa. The expected proportion corresponding is calculated by multiplying and adding marginal frequencies. For example the correspondence for A that can be expected from the marginal frequencies of B is: $8/98 \times 8/98 = 0.007$

The expected proportion corresponding values are:

Code	Marginal frequencies		Expected proportion corresponding
A	$8/98 \times 8/98$	=	0.007
B	$50/98 \times 60/98$	=	0.312
C	$30/98 \times 20/98$	=	0.062
D	$5/98 \times 8/98$	=	0.004
E	$5/98 \times 2/98$	=	0.001
Total			0.386

For the table above Kappa is:

$$\text{Kappa} = \frac{(0.83 - 0.386)}{(1 - 0.386)} = \frac{0.444}{0.614} = 0.72$$

As we see in this example, there is a difference in the outcome of the two measures (0.83 versus 0.72), but the difference is not as large as in the previous example (0.75 versus 0.38). Of course it is hard to say when the Kappa is sufficiently high. We would, generally speaking, say a Kappa should be above 0.70 in order to have an intercoder reliability that is acceptable. If the Kappa is less, we would strongly advise to improve the coding scheme.

7.7 Comparing the coded protocols with the models

7.7.1 Introduction

We now will discuss the stage of protocol analysis that is crucial in the hypothesis testing style, i.e. comparing the coded protocols with the model. If the model is stated in terms of properties of protocols rather than procedures, these properties are measured by counting items or by rating protocols. This requires procedures that are standard in social science research. Comparing *procedural* models with protocols involves a different notion of fit that we shall discuss here.

7.7.2 Comparing protocols with procedural models

Each segment in each of the protocols should fit within the model. In general a model predicts more than one possible process (we called this non-deterministic

models). Each of these corresponds to a possible coded protocol and fitting means verifying if the coded protocol is one of the predicted protocols. You look at the coded segments one by one and compare them with the predicted steps of the model. If a segment (or episode) does not fit into the model because it cannot be generated from the model and the state of the reasoning process then this is marked as a deviation. These deviations are counted. Differences between the coded protocols and the predictions from a process model, may be of three kinds:

1. The protocols show processes which are not predicted by the model.
2. The model predicts processes not shown by the protocols.
3. The protocols show processes in a different sequence than predicted by the model.

1. Unpredicted processes: Unpredicted processes occur when there are segments or fragments in the protocol which cannot be coded, not even under the category 'not task-related or meta-statements'. This means that there is no category in the model for the verbalized actions. For example, in the exploratory phase of Hamel's study, he encountered remarks on the assignment related to how the building should look. These kinds of processes were not at first part of the model and thus were not represented in the coding scheme. What should you decide about your model if you have one or more uncodable segments? The simplest judgement is that your model is false. If the model is detailed and makes strong predictions instead of allowing many possibilities then it is likely to be false. However, usually an interesting question is to what extent the model is false. How much behaviour is covered correctly by the model? Generally speaking, one could say that the more uncoded segments are found, the more evidence there is against the model. This analysis must be done for each element of the model to show which elements are responsible for the discrepancies. Such discrepancies are especially of interest when there are unpredicted processes in protocols from different subjects, and even more so when there exists correspondence between those subjects on the unpredicted processes.

2. Absence of predicted processes: A similar argument applies to the absence of processes that were predicted by the model. If the absence cannot be explained by the verbalization model (as a verbalization failure) then it contradicts the model. The best measure for the absence of predicted processes is simply their number or proportion of the total number of predicted processes.

3. Unpredicted sequences: Sometimes a protocol shows the predicted categories but in a sequence that is not predicted by the model. This applies to process models that contain predictions about sequences of events. Here again unpredicted sequences are evidence against the model and it is necessary to quantify the difference. One method is to take *transitions* as a unit of analysis instead of reasoning steps. A process model may predict possible transitions and unpredicted transitions can simply be counted. This method was used in Hamel's study. In 12 protocols a total of 2829 transitions occurred, 2818 of which were conform the model. Hamel could argue that the remaining deviations were not of such nature that his model should be rejected. Transitions do not take the wider context into account: whether a transition is possible depends on the previous transition and not on any wider context.

A process model usually specifies component processes and a relation between these components. For example, a model that predicts the processes *orientation*, *execution* and *evaluation* can also specify sub-processes of these three. It can now specify that these processes will occur in this order. This means that sequences are predicted at two levels of detail. This gives the same types of errors as we discussed (missing and unpredicted elements and unpredicted sequences) but at different levels. This complicates the analysis. We have no real solution for this problem. A simple method is to report results for each (important) level separately.

7.7.3 Issues in quantifying the fit

There are several issues that complicate quantifying the fit between protocol and model. These are:

(a) Degrees of freedom in the model: A model may not specify a unique description of a protocol, but it may allow several possibilities. These may in turn be dependent on events that occur during problem-solving. For instance, if the subject makes an unpredicted mistake, problem-solving may follow an unpredicted course, but still fit the model. A model allowing many different behaviours is of course weaker, in the sense that it has less predictive power than one that predicts fewer possibilities.

(b) Differences in size of the protocols: Two protocols may differ in length not because of differences in thought processes, but because of different styles of verbalization. This may affect the coding. Usually this effect is reduced by aggregation, because a protocol that is more verbose than another because of the verbalization process, is likely to be the same if phrases are aggregated to problem-solving episodes. If phrases are used then a standardization on the

size of the protocol (by using the percentages) is a good solution.

7.7.4 Comparing sets of protocols

Sometimes groups of protocols are compared, for example sets of protocols of two types of subjects in a psychological experiment or of subjects before and after an experimental treatment. If dimensions are used and protocols can be assigned a score on a dimension, possibly aggregating process properties, then the situation is the same as in standard experiments and the same analysis procedures apply. For example, one may count the proportion (or number) of *orientation* fragments in protocols of novices and experts.

A special problem occurs if the comparison involves process structures. Suppose that we compare beginners and experts in architectural design and predict that experts will follow the order *orientation - execute - evaluate* where beginners will mix the order. How can we quantify the difference between a set of (coded) beginners protocols and a set of (coded) expert protocols? A possibility is to calculate the rank correlation between each protocol and the expert sequence and test the difference between the two groups. However, this procedure suffers from the effect of different numbers of protocol fragments (beginners are likely to need more reasoning to solve the problems than experts). Ideally this effect should disappear when a good coding scheme is used because different ways to verbalize a cognitive process will result in the same code being assigned to a protocol fragment (even if the fragment is longer in one protocol than in another). However, in practice this is not always adequate. The solution followed in practice is to define properties that abstract from these levels on an ad hoc basis.

7.8 Computer support tools for analysis

7.8.1 Introduction

Transcribing and coding or annotating protocols is a very time-consuming activity. Currently, several types of computer systems are being developed to support this task. These systems are still in a stage of development and exploration and they are not yet standardized or easily accessible. Therefore we only summarize the principles of these systems to give an impression of tools that are likely to be available in the near future.

7.8.2 Indexing tools

One useful type of tools are systems that can index and retrieve pieces of text. If protocols are transcribed and stored in the computer, fragments of the text can be marked and given an index. These indexes can again be stored. For example, we can give each fragment an individual reference but collect all references of fragments where the protocol concerns, for example, 'selecting a schema'. This makes it possible to quickly retrieve all fragments where this occurs. This type of tool is relatively easy to build. It can be constructed using most hypertext and database systems. The structure of the database is to use segments or aggregated fragments of the written text as entries in the database and to index these by coding category, coder, date (of collecting the protocol), problem (that is the problem solved here) and subject. Future technical developments will make it possible to store the spoken protocol and index it in this way, thus avoiding transcription and keeping additional information in the spoken protocol.

7.8.3 An implemented model as tool

The most powerful tools are those that correspond to implemented psychological models that can be used to recognize uncoded or coded protocols. The model of physics problem-solving described in Chapter 8 has been implemented. The resulting system can actually solve most of the problems that were used in the study described there. A recognition mode was added to this system as follows. In recognition mode, the system is given the problem and then it asks the user of the system (who has the protocol), what the next lines in the protocol are. The answer must be given in a predefined form. The system then checks if this is consistent with what it expects. If this is the case, it continues. If the protocol deviates from the expected solution path, this is noted as an *unpredicted* step (which violates the model). In many cases the system can understand the step taken in the protocol and resume problem-solving from the new situation. In this way protocols can be coded quickly and an overview of unpredicted and missing events can be produced automatically.

7.9 Reporting the results of protocol analysis

Analysis of think aloud protocols usually leads to a large amount of documentation: the typed protocols, the detailed model, the coded protocols, the comparison between the model and the coded protocols. Technical reports on