	self-initiated explanation – inference – assumption
	self-initiated explanation – inference – self-initiated explanation
	self-initiated explanation – inference – evaluation
	self-initiated explanation – inference – decision
	self-initiated explanation – evaluation – inference – self-initiated explanation
	self-initiated explanation – query – self-initiated explanation
	self-initiated explanation – search – inference
	self-initiated explanation – inference – conflict
	self-initiated explanation – evaluation – assumption
	self-initiated explanation – assumption
Assumption	assumption – inference
	assumption – inference – assumption
	assumption – inference – self-initiated explanation
	assumption – inference – evaluation
	assumption – self-initiated explanation – conflict/intention
	assumption – conflict
	assumption – conflict – evaluation
	assumption – action
	assumption – intention
	assumption – search
Conflict	conflict – inference – self-initiated explanation
	conflict – inference – search
	conflict – evaluation – search – evaluation
Action	action – self-initiated explanation
Task	task – assumption
	task – search
Decision	decision – assumption
	decision – inference
	decision – inference – intention
	decision – evaluation

11 The Data in Design Protocols: The Issue of Data Coding, Data Analysis in the Development of Models of the Design Process

Terry Purcell, John Gero, Helen Edwards and **Tom McNeill**
University of Sydney, Australia

A video/audio protocol of a design session represents a particular form of qualitative data. More typically qualitative data takes the form of verbal data such as a record of the answers to open-ended questions, transcripts of relatively unstructured group discussions or existing text from one or more sources. However, while protocols represent a particular form of qualitative data, they share with the other forms the fact that the protocols, transcripts and so on are not the actual data on which analysis is performed. Rather the protocol or transcript has to be segmented or categorized in some way with the frequency of occurrence of each category forming the data that are interpreted and discussed. Design protocols are also different in that most of the more traditional forms of qualitative data do not have time as such an important facet of the data.

In many of the areas in which qualitative data have been gathered and analysed, the traditional way in which the original

data have been segmented is by inferring the categories by a careful reading of the original record or part of the record. The frequencies of occurrence of the categories are then obtained by a reanalysis of the full record. A more recent and potentially richer approach to the analysis of qualitative data is the 'grounded theory' approach of Glaser and Strauss[1]. Here the data are not segmented with each segment then being classified into a single category. Rather the original record is repeatedly revisited with both multiple codings of the same segment possible and with differing sets of segments of finer or coarser grain being recognized.

An examination of the design protocol literature indicates that it is the traditional approach to data segmentation, analysis and interpretation which has formed the basis for much of the initial work in the area. However, more recently, there has been a shift in the analysis of design protocols towards imposing an externally derived structure onto the protocols, perhaps to a greater degree than in the traditional areas where qualitative data have been collected and interpreted. This external structure has been based on existing analyses and models of the design process[2–6]. Segmentation of the original protocol in this approach is not generated by the data but based on other criteria, such as pauses in the flow of words, or on semantic/syntactic criteria for recognizing discrete utterances. There are two consequences of this shift. One is that the number of categories used in the analysis is relatively limited. For example, in the recent paper by Lloyd and Scott[5] three categories are used – generative, deductive and evaluative utterances. While this particular experiment and others using this approach have produced interesting results, they represent a severe loss of detail from the original protocol. The second is that the results of the later experiments are difficult to compare to the earlier work and to each other. As a result, while the intention underlying the shift to imposing structure on protocols was to test existing hypotheses and models of the design process, the wide variation in the categorical structures imposed actually inhibits comparisons between experiments and the development of theory. However, a reading of this literature also reveals commonality and overlap between the categories used and tantalizing possible correspondences between the results that have been obtained.

This analysis of the nature of the data collected in a protocol study of design has formed the starting point for our analysis of the protocol of the session involving an individual designer. What we have attempted to develop is a coding scheme that brings together these various approaches and addresses the problems we have identified. First, the coding scheme embodies both data and theory generated categories. The data generated categories are derived from previous work and from this particular protocol. While these categories depend on the data, it was also apparent that many of them were similar to the more theoretical categories employed in recent work in the area. The theoretical categories are derived from the work of Gero *et al.*[7–9] where it is argued that design involves reasoning in three domains – function, structure and behaviour. Second, the coding scheme that we have developed is far richer than that used in most previous work. For example, Eckersley[4] developed a coding scheme that consisted of eight categories, a number which considerably exceeds that used in previous work. In contrast, our coding scheme consists of 27 categories. The danger in increasing the number of categories is that the results become too complex, potentially masking relationships and patterns in the data, and are generally too difficult to understand. While this presents problems, we have attempted to counter them by the structure that is present in the coding categories and by the methods that are used to visualize the results. Finally, while time is an important facet of the analysis of a design protocol, as noted previously, we consider that too great an emphasis on changes that occur over the period of a full design session mask significant aspects of what is occurring either simultaneously in the sense of the same segment having multiple codings or in relationships between segments; that is at a much smaller time scale than that involved in a full design session. This paper therefore reports the process of development of our coding scheme and presents a number of examples of the application of this scheme which illustrate the results of our approach.

1 The Development of the Coding Scheme

The current coding scheme was developed by Gero and McNeill[10] over a number of iterations from a version previously used to analyse conceptual electronic design sessions. Early work carried out by McNeill and Edmonds[11] concentrated on the problem domain and the designer's navigation through the

problem domain. This meant that, by necessity, the analysis was problem specific and results were dependent on the particular problem being tackled by the designer. The work centred around the notion of attaching labels to events within the design episode and viewing the design episode as a sequence of inter-dependent events.

In subsequent work by Gero and McNeill[10], it was desirable to compare a number of design episodes and the need was seen for an approach that would give a result that was problem independent. The coding scheme which they developed evolved over a period of time with each iteration being examined for how well it captured significant characteristics of the design episodes. As the coding progressed new codes were added to describe events that were not appropriately covered by an existing code. The resulting scheme incorporated three broad classifications, two of which were concerned with the designer's place in the problem space while the third was concerned with the designer's activity at any time.

The coding scheme used in the analysis of the Delft individual design session was a modified version of the scheme developed by Gero and McNeill. Although the scheme was considered to be problem independent, the coders first perused the Delft individual design session in an unstructured way in order to ensure that the coding scheme would be appropriate. An important difference in experimental procedure was revealed between the Delft individual design session and the electronic engineering sessions on which the coding scheme was based. The electronic engineering sessions featured designers working on tasks that they were required to complete as part of their own everyday work. As a result, most of the work of defining and analysing the problem had already been carried out prior to the recorded design sessions. In the Delft experiment, the designer was given a design problem to which he had not previously been exposed. From the perusal of the Delft tape, it was apparent that in order to complete the task, the designer would need to analyse the problem to a greater extent than the designers in the electronic engineering sessions. The coding scheme was then revised in order to take this difference into account. The coding scheme as used in the present analysis of the Delft individual design session is presented in Table 11.1 and discussed in greater detail in the following section.

2 A Description of the Content of the Coding Scheme

The essence of the coding scheme is that each event in a design episode is categorized over three broad classifications. The first two classifications centre around the problem being solved. The third of the three classifications encoded is concerned with the designer's activity at any time and provides a problem-independent event analysis of the data. Details of the codes which constitute each of the three classifications are outlined in Table 11.1.

The first classification, *Problem Domain: Level of Abstraction*, is established as the coding progresses. This group of codes is related to the way the designer appears to be dissecting the problem as well as a reflection of the complexity of the problem itself. In the present design episode, the *System* level (0) relates to how the finished product would be used and the relation between the bicycle, backpack and backpack carrier as a whole assembly. The bicycle, backpack and backpack carrier could all be considered as subsystems in this design episode. The *Subsystems* level (1) indicates that the designer is focusing on issues which pertain to only one of these subsystems at that particular time. The *Detail* level (2) indicates that the designer is considering the characteristics of physical components.

The second classification, *Problem Domain: Function, Structure and Behaviour*, monitors the forms of reasoning used throughout the design episode. Reasoning about *Structure* (S) involves the manipulation of physical properties in order to generate a physical solution to an abstract problem. Reasoning about *Behaviour* (B) concerns the description of the object's actions or processes in given circumstances. Reasoning about *Function* (F) refers to the manner in which the designed object fulfils its purpose. The 'R' code was used to highlight when the designer was establishing or revising his or her requirements.

The third classification, *Strategy* describes the moment-to-moment activities of the designer during the design episode. The codes used can be dissected into four groups. The first group is concerned with analysis of the problem. The second group is concerned with the proposal of a solution and the third is concerned with analysis of the proposed solution. The fourth group is concerned with times when the designer made explicit reference to what he or she was doing or what he or she knows. An additional 'X' code has been included to indicate occasions when the experimenter interacted with the designer or made a comment. Some examples of how these codes were applied to the Delft individual design session are listed in Table 11.2.

Table 11.1 The coding scheme

First Classification – Problem Domain: Level of Abstraction

Describes where the designer is within the problem domain in terms of level of abstraction.

The code consists of a numeral. The numerals indicate different aspects of the problem domain.

0 – System	The designer is considering the problem from the point of view of the user.
1 – Subsystems	The designer is considering the problem in terms of the subsystems.
2 – Detail	The designer is considering the details of the subsystems.

Second Classification – Problem Domain: Function, Structure and Behaviour

The letters indicate various aspects of the problem domain.

F – Function	The designer is working with the functional aspects of the problem domain.
B – Behaviour	The designer is working with the behavioural aspects of the problem domain.
S – Structure	The designer is working with the structural aspects of the problem domain.
R – Requirements	The designer is adding to, modifying or reconsidering aspects of the initial requirements.

Third Classification – Strategy

Describes the designer's immediate actions and his short term strategy. This classification is divided into four groups of codes. The list includes examples to illustrate the meaning of the code.

Analysis Problem

Ap – Analysing the Problem	'What is the system going to need to do....'
Cp – Consulting Information about the Problem	As above but using external information
Ep – Evaluating the Problem	'That's an important feature....'
Pp – Postponing the Analysis of the Problem	'I can find that out later....'

Table 11.1 *(continued)*

Synthesizing Solution

Ps – Proposing a Solution	'The way to solve that is....'
Cl – Clarifying a Solution	'I'll do that a bit neater....'
Re – Retracting a Previous Design Decision	'That approach is no good what if I....'
Dd – Making a Design Decision	'OK. We'll go for that one....'
Co – Consulting External Information for Ideas	'What are my options....'
Pd – Postponing a Design Action	'I need to do ... later'
La – Looking Ahead	'These things will be trivial (difficult) to do'
Lb – Looking Back	'Can I improve this solution?' 'Do I need all these features?'

Evaluating Solution

Ju – Justifying a Proposed Solution	'This is the way to go because....'
An – Analysing a Proposed Solution	'That will work like this....'
Pa – Postponing an Analysis Action	'I'll need to work that out later'
Ca – Performing Calculations to Analyse a Proposed Solution	'That's seven inches times three'
Ev – Evaluating a Proposed Solution	'This is faster, cheaper etc....'

Explicit Strategies

Ka – Explicitly Referring to Application Knowledge	'In this environment it will need to be....'
Kd – Explicitly Referring to Domain Knowledge	'I know that these components are....'
Ds – Explicitly Referring to Design Strategy	'I'm doing this the hard way....'
X – All comments made by the experimenter	

Table 11.2 Examples of statements from the protocol coded according to the Strategy classification

Code	Examples from the protocol
Analysing Problem	
Ap – Analysing the Problem	'...and first of all I notice that it's pretty small but the product spec wasn't that specific...'
Cp – Consulting Information about the Problem	'is Batavus willing to add any fittings to their bicycle in their production line to accommodate this product?'
Ep – Evaluating the Problem	'that's a pretty heavy backpack'
Pp – Postponing the Analysis of the Problem	'I can calculate that and I'll do that later'
Evaluating Solution	
Ju – Justifying a Proposed Solution	'another reason I make this in aluminum is because bike works are good at welding'
An – Analysing a Proposed Solution	'any bends these rods are then going to be in tension and compression'
Pa – Postponing an Analysis Action	No equivalent example
Ca – Performing Calculations to Analyse a Proposed Solution	'forty times one inch is point nought four cubic inches per inch'
Ev – Evaluating a Proposed Solution	'certainly I would say that that sorta looks classy having a backpack in the centre....'
Synthesizing Solution	
Ps – Proposing a Solution	'immediately my impression is hey it's nice to have ... putting it on the front handlebars'

Table 11.2 (*continued*)

Code	Examples from the protocol
Cl – Clarifying a Solution	'OK so what we have is, we have a, let's see if I can draw a sketch here'
Re – Retracting a Previous Design Decision	'we're going to have to scrap this'
Dd – Making a Design Decision	'so that's one of the possible positions'
Co – Consulting External Information for Ideas	'these we know are designs that the chap had worked out'
Pd – Postponing a Design Action	'I can't just remember right now think about that em er'
La – Looking Ahead	'actually what I would do is run down to the local windsurfing store and buy a boom extension and I would mock it up....'
Lb – Looking Back	No equivalent example
Explicit Strategies	
Ka – Explicitly Referring to Application Knowledge	'having used a backpack on a bike in the past ... I learned very early on that....'
Kd – Explicitly Referring to Domain Knowledge	'I know that em the issue is going to be bending stiffness'
Ds – Explicitly Referring to Design Strategy	'OK it's five-ten I've gotta start getting to a more concrete design'

3 Method

All forms of qualitative research are based on subjective judgements about the data by the coder(s). As a result, there can be variation between individuals who code the same set of data. A common means of demonstrating the accuracy of the results is

to utilize more than one coder and to quantify differences in their interpretation of the data by calculating inter-rater reliability. For example, where a protocol contains a set number of standardized units and each unit is assigned a particular code, inter-rater reliability represents the number of units for which the coders assigned the same code, as a proportion of the total number of units.

However, this is not the process used in the current analysis of the Delft individual design session. Rather, the process used to code the present design session relies on the division of the protocol primarily according to meaning, rather than standardized units, and so there is potential for variation between coders in the number and length of units identified, as well as differences in how these units were coded. Under these circumstances, the suitability of inter-rater reliability as a means of ensuring accurate coding is questionable. As a result, we adopted an alternative strategy to ensure that we achieved an accurate analysis of the design session. The strategy we adopted acknowledges differences in interpretation between coders and views these differences as a legitimate opportunity to explore the data in greater detail and to develop a coherent, consensus coding which reflects the structure in the data.

This strategy is based on the Delphi Method which may be characterized as a method of structuring group communication processes in order to achieve agreement[12]. A series of codings was conducted by two independent coders, with an arbitration process providing the means to resolve any differences in the codings in order to converge towards a coherent final coding. The strategy involved a four-stage process of coding. The two coders worked independently in the first three stages. In the first stage, each coder applied the coding scheme to the design episode. The process for applying the coding scheme is described in greater detail below. After a period of time (several days) the episode was coded a second time. The period of time allowed for a more independent second coding. After a further period of time, each coder carried out the first arbitration process, which was to compare their own two codings. Each coder worked independently to produce a third coding in which any differences between their first two codings were resolved. In the final stage, the arbitration between coders, the two coders worked together to arbitrate between the results they had each produced.

3.1 Some Background Information on the Two Coders

The first coder was experienced in the use of the coding scheme while the second coder was trained in the application of the scheme for this workshop. Training consisted of a number of discussions concerning the coding scheme as well as some practice. The second coder practised applying the coding scheme on 20-minute sections of two different videos. One video was of an electronic engineering design episode while the other was of the Delft individual design session. The results obtained by the experienced coder for the same sections of the videos were compared and used as a guideline for the revision of coding strategies.

3.2 The Coding Process for Individual Coders

This section describes in greater detail the process which each coder followed, while working independently, to apply the coding scheme to the Delft individual design session. The video, written transcript and the copies of the designer's sketches were all utilized by the two coders to guide their interpretation of the video. The video was viewed a number of times in order to increase familiarity with the experimental situation and the overall sequence of events. The actual coding was recorded on the written transcript. The transcript was formatted in a way which would provide a legible record of the identified segments and the codes which were applied to these segments. Specifically, the transcribed speech was formatted into a narrow column so that each line contained an average of six words. The format of the transcript is represented in Figure 11.1. Three blank columns, which correspond to the three broad classifications in the coding scheme, were positioned to the left of the text column. The codes were handwritten in these columns. Each coder, working independently, went through the whole episode and identified segments to which they applied a particular code from each of the three classifications. An exception to this procedure was that, where a segment was identified as an explicit strategy (the fourth group of codes in the classification *Strategy*), that segment was not coded according to the other two classifications. This was done because by nature, the explicit strategies do not have any bearing on the designer's consideration of the problem domain. The beginning and end of the coded segments were marked on the transcribed speech with a slash. When a certain code in any column is applied to a particular part of the episode, the speech or activity which follows is then examined in order to determine if the same code applies,

			00:56:00
1	S	Ps	side mounted alright /obviously
1	B	An	we might wanna put the backpack on this (mutter) like that oh let's see there's another way out here huh ah that's gonna give some stability issues here we have some stability issues here we have some stability
1	S	Ps	issues there em em /OK er we
1	B	An	might be able to em do this /I think I'd be worried about the pack sorta not really being well suited there (mutter) is facing in
		Ds	em top down /OK I have spent er fortyfive minutes or so now forty minutes doing that er so em ... (mutter) going to check every possible location
			00:57:00

Figure 11.1 The transcribed speech has been formatted into a narrow column so that handwritten codes can be aligned with the corresponding segments. The beginning of each new segment is indicated by a slash.

or if it should be considered as a separate segment with a different code. A new event is associated with a change in code in at least one column. In some cases, new segments are associated with a change in all three columns, while in other cases, the code recorded in a particular column may simply be repeated from the preceding segment. Where a code is repeated, this indicates that the particular code applies for the duration of that series of segments.

3.3 Arbitration between Coders

In order to facilitate the arbitration process, both individual coders' results were added to the transcript, along with a third set of blank columns to record the arbitrated coding. As a result of the coding process used, in which each coder worked independently in dividing the design episode into segments, there can be multiple codes of a single segment of the design episode or the same code can persist over a number of segments. Therefore, for the purpose of arbitration between the coders, it was necessary to segment the design episode according to both codings. In doing so, wherever a segment had been identified by either coder, this became a new segment for the purpose of arbitration. This is demonstrated in Figure 11.2. A total of 575 segments were arbitrated between the two coders. The first step

FIRST CODER			SECOND CODER			ARBITRATION			TRANSCRIBED SPEECH
0	B	An	2	B	An		B	An	... alright so OK obviously if they cut loose the wheel would come loose too so we're not gonna worry about that too much
2	S	Cl	2	S	Ps	2	S		OK so we're gonna have to have these rear stays here go down to lugnuts here to lugnuts stay to lugnut alright alright and then this is this is going to ...
2	S	Re	2	S	Re	2	S	Re	em I don't like that 02:00:00
2	S	Ps	2	S	Ps	2	S	Ps	what we're going to do is we're going to run these stays to em to this er this fitting here
2	B	An	2	B	An	2	B	An	we wanna run this to here so that we don't have to rely on the lugnuts em unfortunately the angle at which this is gonna be is gonna be a function of the em bicycle so this is gonna have to be at different angles here too
2	S	Ps	2	S	Ps	2	S	Ps	so we're gonna have a pivot through there OK i got that figured out em we em er got to have a pivot here
			2	B	An				don't like that pivot here em don't like em that
			2	B	An				but unfortunately this is just this is just handling small loads like this tipping loads em

Figure 11.2 Transcript used for purpose of arbitration. Lines across the columns and transcribed speech indicate each segment. Identical codes were recorded in the arbitration columns.

in the arbitration process was to identify those segments where identical codes were given by the two coders. In identifying identical codes for each segment, each of the columns was treated separately, that is, an identical code could occur in any of the three columns and not necessarily all three. These identical codes were recorded into the arbitration columns; where the codes were not identical in any of the columns, that column was left blank and the final decision was added following arbitration.

A number of types of differences in codes were observed, all of which were amenable to discussion and arbitration. The first

source of differences is similar to that of traditional qualitative analysis, in that the coders had applied different codes to the same segments. In these cases, each coder provided a justification for their choice with reference to the source material and an agreement was then reached. In some cases, it was decided to adopt the code applied by one of the coders; at other times a completely new code was applied. The other sources of differences involved only partial disagreement. On some occasions, although the two coders applied the same codes to a particular segment, the demarcation of that segment by the two coders differed slightly, usually only by a few words. Another source of differences in coding was when the two coders had segmented the transcript at different levels of detail. For example, one coder may have identified one large segment while the other broke this down into a number of smaller segments. Sometimes through discussion, the two coders decided that the greater level of detail proposed by one coder was not warranted. However, on other occasions, the finer detail was deemed appropriate and was added to the final code. This is an example of the way in which arbitration can be used to establish a more coherent and detailed final coding. In comparison, more traditional approaches may result in a loss of information from the original protocol. A single coder may overlook some of the finer detail. Where more than one coder is employed, the aim is usually to demonstrate that a high degree of similarity between codings can be achieved, rather than to synthesize the results into a richer interpretation of the protocol.

To achieve the final coding decision for each segment, both coders worked together to discuss and examine analytically the underlying reasons for the differences in codes, by referring either to particular parts of the transcribed speech or activity observed in the video or to previous coding decisions. The transcript was then reorganized according to the segments defined in the arbitration. The timer facility in the video was used to log the times associated with each event. An exemplary section of the final coding produced by the arbitration is presented in Appendix A.

4 Results and Example Analyses

Figures 11.3 and 11.4 illustrate how our coding scheme can represent events in a design protocol as a function of time over a full design session. The figures also demonstrate how a

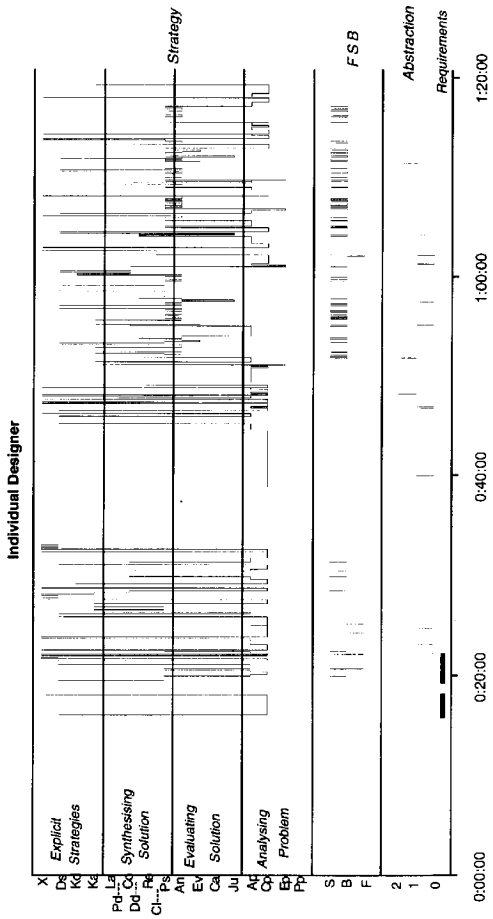

Figure 11.3 Coded events as a function of time for the first hour of the design episode.

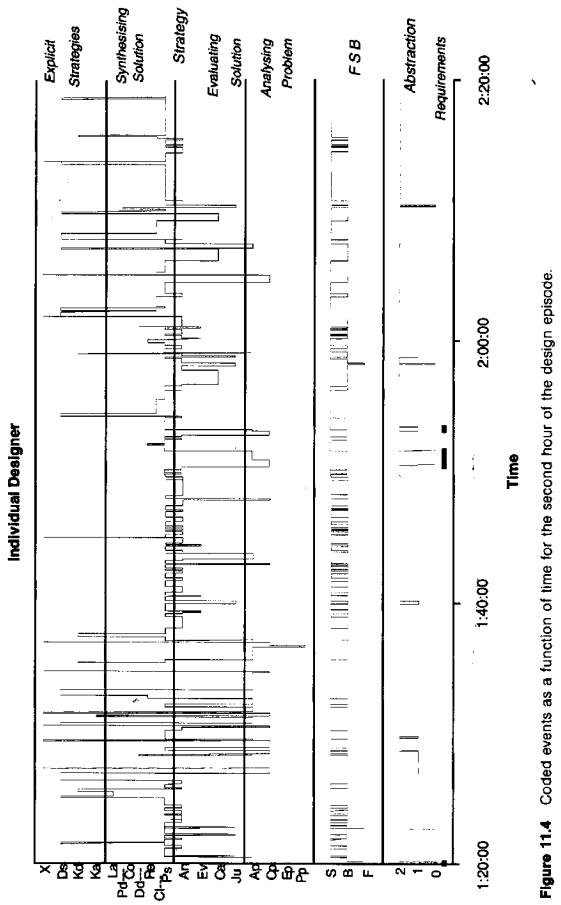

Figure 11.4 Coded events as a function of time for the second hour of the design episode.

complex set of coding categories can be visualized and how this facilitates examining the relationship between the categories. The design session has been divided into one-hour time segments. Figure 11.3 represents the first hour of the session and Figure 11.4 represents the second hour. Together, the two figures provide an overview of the designer's activity throughout the design session, in terms of the 27 categories used in the coding scheme.

For each of the classifications *Strategy, Problem Domain: Function, Structure and Behaviour* and *Problem Domain: Level of Abstraction*, the pattern of vertical and horizontal lines portrays the designer's activity at any particular time. The vertical position of the line corresponds to the relevant code and the horizontal position of the line indicates the amount of time associated with that code. For example, the upper portion of Figure 11.3 indicates that the first event in the design episode has been coded as an explicit reference to a design strategy (Ds), followed by a period of analysing the problem (Ap). By examining the lower part of the figure, it can be seen that this analysis of the problem corresponds to the designer being engaged in reasoning about Function (F) at the highest level of abstraction, the system level (0). The designer also considered the initial requirements at this time. In the figures, there are some quite noticeable breaks in the line. These breaks occur for two reasons. Firstly, some sections of the design episode pertain only to the experimental situation and not the design process and so these sections were not coded. The large break which occurs between 00:35:00 and 00:40:00 corresponds to the time when the designer made a phone call and was waiting to get through to the relevant person. The second reason is that explicit strategies are not associated with the two problem domain categories, and so, whenever an explicit strategy occurs, a break appears in the lines which represent the *Problem Domain: Function, Structure and Behaviour* and *Problem Domain: Level of Abstraction* categories.

As well as providing information about the designer's activity at any time, the figures also allow the reader to discern patterns of activity across the design episode and comparisons across the three classifications can also be made. A brief interpretation of the design episode derived from the figures is outlined below.

Overall, the designer utilizes four main strategies throughout the design episode, Proposing a solution (Ps), Analysing a

proposed solution (An), Analysing the problem (Ap) and Consulting information about the problem (Cp). The pattern observed in the first half of the design session clearly demonstrates that the designer is building an understanding of the problem (see Figure 11.3). The longest period of time associated with any one strategy occurs between 00:39:54 and 00:44:28, when the designer uses a telephone conversation as a means of consulting information about the problem. Indeed, the designer spends a considerable amount of time consulting information about the problem (Cp) and analysing the problem (Ap) until around 1:40:00, after which this type of activity gradually decreases. This coincides with an increase in the amount of time engaged in synthesizing a solution (see Figure 11.4). In the latter two-thirds of the session, the designer proposes and then analyses a solution in a rapid series of cycles. In the final stages of the design session, the designer concentrates almost exclusively on proposing solutions.

In terms of *Problem Domain Function, Structure and Behaviour* the figures indicate that a greater amount of functional reasoning occurs at the beginning of the design session (until around 00:35:00) relative to the latter half. A return to function corresponds to the highest level of abstraction (0). A pattern which is repeated throughout the design episode is that of a period of rapid changes between reasoning about behaviour and structure followed by a period of slower changes. The periods of rapid changes appear to be associated with synthesising and evaluating solutions at the lowest level of abstraction while the slower periods seem to coincide with analysis of the problem.

Unlike the changes observed in *Strategy* and *Problem Domain Function, Structure and Behaviour*, the patterns observed in relation to *Problem Domain: Level of Abstraction* reflect a less rapid rate of change. The designer focuses on a particular level of abstraction for quite lengthy periods of time. The overall pattern suggests that the designer steadily decomposes the problem, beginning at the system level (0), then considering the subsystems (1), and finally addressing the detail level (2). Occasionally, the designer switches back from detail to the system level, but not from detail to subsystems. The designer mostly considers the requirements of the problem at the very beginning of the design session; however, he also addresses the requirements towards the end of the session, prior to an increase in reasoning about structure.

Figures 11.5 and 11.6 illustrate how theoretical predictions can be tested using this approach. Gero *et al*[7–91] have argued that design involves three types of reasoning – reasoning about function, about behaviour and about structure. Furthermore, this view proposes that design generally proceeds from a conceptual description of a problem (which involves reasoning about function and behaviour) to a description of an artefact as a solution to the problem (this involves reasoning about structure). A simple specification of this hypothesis is to examine the percentage of the time spent using these three types of reasoning over a design session. In order to explore this hypothesis, the design episode was divided into one-minute intervals. For each interval, the percentage of time spent by the designer in reasoning about Function and Behaviour (as compared to Structure) was calculated. Using a spreadsheet program, graphs were generated which indicate changes in the proportion of Function/Behaviour to Structure over the duration of the design episode. Using this method, it is possible to alter the time scale of the graph. Since the average event in the design session is around half a minute long and some events are longer than one minute, a graph which consists of one-minute intervals tends to oscillate from zero to 100% and this obscures any pattern in the data. To discover if patterns do occur and reduce the effect of noise in the data, moving averages were calculated over five- and ten-minute intervals. That is, at each one-minute interval, averages were calculated over the preceding five- or ten-minute period.

Figure 11.5, which presents the percentage of Function and Behaviour codings averaged over 10-minute time intervals, represents the overall trend in the data. This figure demonstrates a very systematic decline in these two types of reasoning over the full design session, indicating the increasing importance of structural reasoning. The design session begins with 100% Function/Behaviour and finishes with 100% Structure. Further it is apparent that there are two broad rates of decline in these two types of reasoning. The first is relatively slow and occurs throughout the majority of the design session. The second is very rapid and takes place at the end of the session. However, it is also apparent that within this overall trend there are systematic variations – that is periods where structural reasoning declines and the other two types of reasoning show a relative increase. The systematic nature of these variations

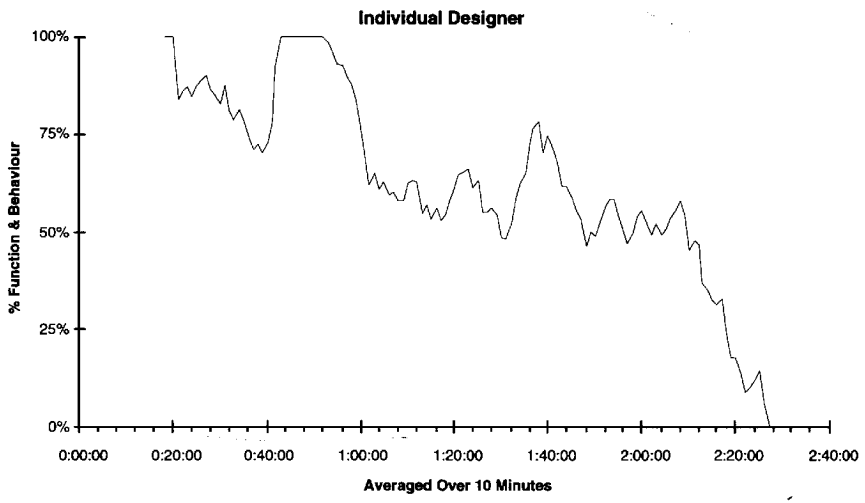

Figure 11.5 The percentage of time spent in reasoning about Function and Behaviour as compared to Structure, calculated at one minute intervals and averaged over periods of ten minutes.

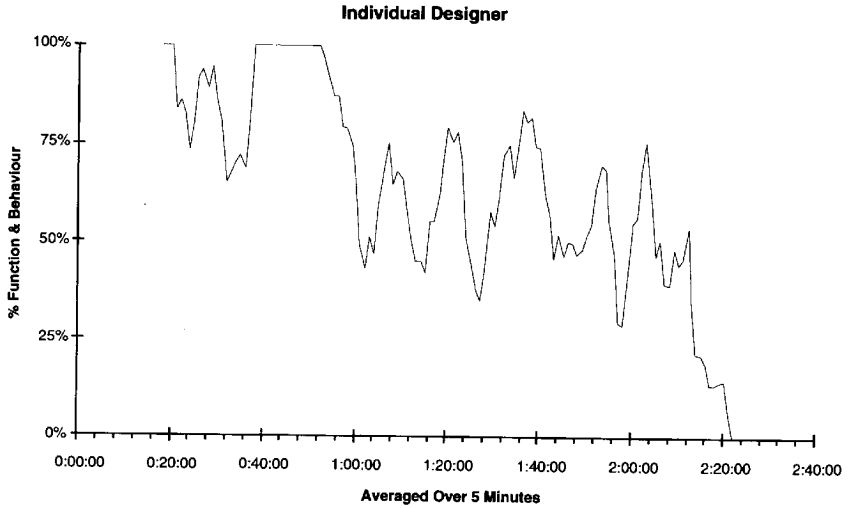

Figure 11.6 The percentage of time spent in reasoning about Function and Behaviour as compared to Structure, calculated at one minute intervals and averaged over periods of five minutes.

within the overall trend can be demonstrated through the finer-grained analysis provided by the five-minute moving averages shown in Figure 11.6. From this graph it is apparent that this variation within the overall trend is very systematic with a series of clear cyclic changes present, particularly in the second half of the session. It is also apparent that the rapid decline in these forms of reasoning at the end of the session is essentially unaltered in the finer-grained analysis.

If moving averages over a five-minute period are calculated separately for Function, Structure and Behaviour (Figure 11.7), another significant aspect of the design session becomes apparent. The amount of reasoning about Function is quite small overall and occurs as a very high percentage at the very beginning of the session with little of this type of reasoning subsequently. This is immediately followed by a rapid rise in reasoning about Behaviour which essentially dominates over the next 40-minute period. This is then followed by a rapid rise in the amount of reasoning about Structure with the succeeding period consisting of cycles between reasoning about Behaviour and Structure. Finally these cycles of different types of reasoning are quite abruptly replaced by a rapid decline in reasoning about Behaviour and a rapid increase in reasoning about Structure as the design is finalized.

In terms of the Gero *et al.* model[7–9], these analyses reveal that the first part of the design session does consist of conceptual reasoning as predicted. However the two types of conceptual reasoning effectively occur independently of each other with Function occurring at the very beginning of the session followed by Behaviour. Reasoning at the conceptual level also does not only occur at the beginning of a design session but a particular form of conceptual reasoning, reasoning about Behaviour, persists until very near the end of the session. While these patterns are based on one design session involving one designer, they are potentially theoretically significant and demonstrate how the approach developed here can be used to assess theoretical positions.

These patterns clearly pose the question of what is actually happening when these cycles are occurring. Figure 11.8 is a five-minute moving average based on the sets of items in the Analysis, Synthesis and Evaluation coding categories (see Table 11.1). Comparison of Figures 11.7 and 11.8 demonstrates that the patterns in the two graphs are closely synchronized.

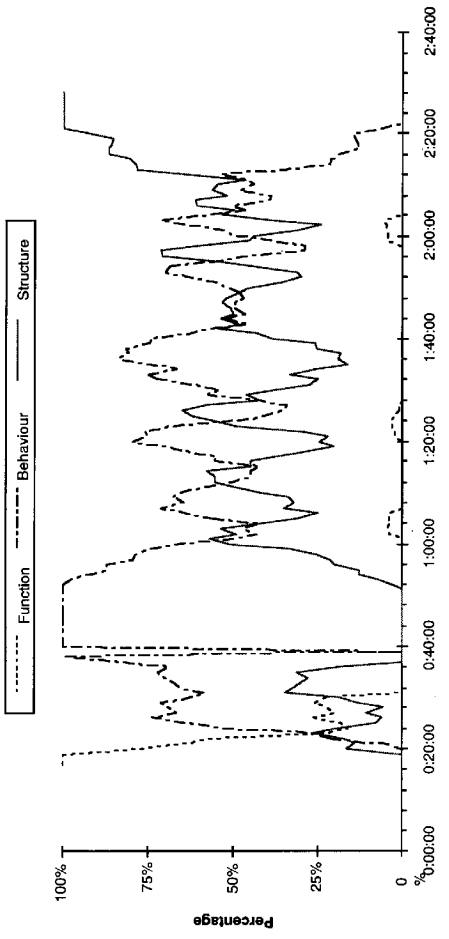

Figure 11.7 The percentage of time spent in reasoning about Function, Behaviour and Structure, calculated at one minute intervals and averaged over periods of five minutes.

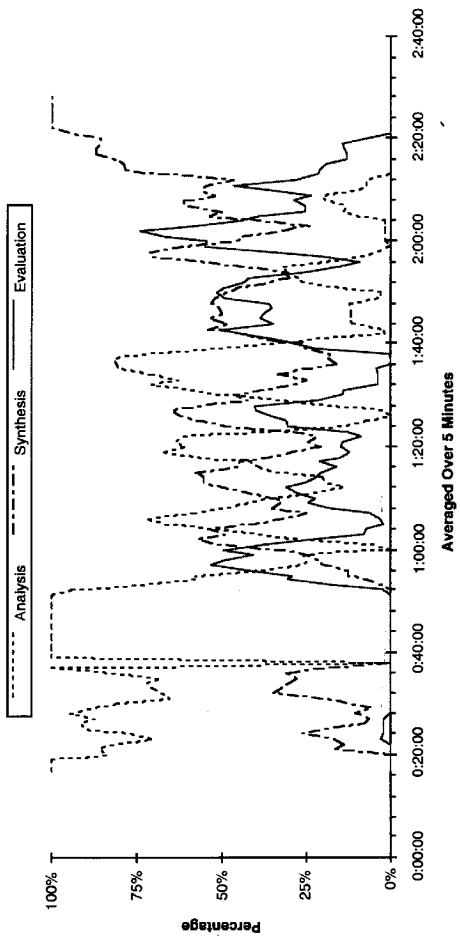

Figure 11.8 The percentage of time spent in Analysis, Synthesis and Evaluation, calculated at one-minute intervals and averaged over periods of five minutes

Analysis dominates the first part of the session when the designer is reasoning about Function and Behaviour. There is then a similar change to that which occurs in the Function. Behaviour, Structure graph. Increases in Analysis are followed by increases in Synthesis and Evaluation. These cycles correspond closely to the Behaviour, Structure cycles indicating that Behavioural reasoning is associated with Analysis and Evaluation while Structural reasoning is associated with Synthesis.

This more detailed analysis in conjunction with the Function, Behaviour, Structure analysis also allows an evaluation of a number of the other theoretical positions that can be found in the literature. For example, Lloyd and Scott[5] identify an engineering model of design based on analysis of problems followed by the synthesis of solutions and an architectural model of design where solutions need to be generated so that the problem can be analysed and understood. The patterns demonstrated in this design session indicate that, if they occur with other designers and with different problems, a more complex model is needed. Analysis clearly plays a central role but that role does not simply involve a period of analysis at the beginning of a design session. Rather analysis dominates at the beginning of the session with the analysis involving both reasoning about Function at the System level and about Behaviour at the Subsystem level. Analysis about Behaviour, however, continues after a marked drop in reasoning about Function and later in the session enters a systematic pattern of relationships with reasoning about Structure at the level of Details about Subsystems which is also associated with Synthesis and Evaluation. This part of the overall record of the design session is therefore consistent with the architectural model of the design process. Examination of the details of the protocol also reveals that the cycles of Analysis and Synthesis/Evaluation do not appear to be consistent with a view that argues that the designer breaks down the initially ill-defined problem into a series of well-defined problems which are then solved through an analysis followed by a synthesis process. Rather the cycles appear to governed by shifts in attention rather than a systematic reasoning process. The approach to the analysis of protocol data we have developed therefore seems to provide a way of developing more detailed theoretical models of the design process and would appear to be sufficiently sensitive to allow the examination of some of the particularly significant issues

which have been appearing in the literature. Such issues as the differences between students and experts, between experts with differing types of training, and between problems which are typical and atypical for a discipline, can all be assessed by detailed comparisons of protocols obtained with the appropriate combinations of different groups of designers and different types of problems.

References

1 Glaser, B.G. and A.L. Strauss, *Discovery of Grounded Theory: Strategies for Qualitative Research*, Aldine, Chicago (1967)
2 Akin, Ö. A structure and function based theory for design reasoning, in N. Cross, K. Dorst and N. Roozenburg, (Eds), *Research in Design Thinking*, Delft University Press, Delft (1992)
3 Christiaans, H. and K. Venselaar, Practical implications of knowledge-based design approach, in N. Cross, K. Dorst and N. Roozenburg (Eds), *Research in Design Thinking*, Delft University Press, Delft (1992)
4 Eckersley, M., The forms of design processes: a protocol analysis, *Design Studies* **9**(2) (April 1988) 86–94
5 Lloyd, P. and P. Scott, Discovering the design problem, *Design Studies*, **15**(2) (April 1994) 125–140
6 Tzonis, A., Huts, ships and bottleracks: design by analogy for architects and/or machines, in N. Cross, K. Dorst and N. Roozenburg, (Eds), *Research in Design Thinking*, Delft University Press, Delft (1992)
7 Gero, J.S., Design prototypes: a knowledge representation schema for design, *AI Magazine*, **11**(4) (1990) 26–36
8 Gero, J.S., K.W. Tham and H.S. Lee, Behaviour: a link between function and structure in design, in *Preprints IntCAD '91*, IFIP, Ohio State University, Columbus (1991)
9 Rosenman, M.A. and J.S. Gero, The what, the how and the why in design, *Journal of Applied Artificial Intelligence* (1994) (in press)
10 Gero, J.S. and T. McNeill, A method for the protocol analysis of design sessions submitted to *Research in Engineering Design* (1995).
11 McNeill, T. and E.A. Edmonds, Empirical study of conceptual electronic design, *Revue Sciences et Techniques de la Conception* **3**(1) (1994) 61–86.
12 Linstone, H.A. and M. Turoff, (Eds), *The Delphi Method: Techniques and Applications*, Addison-Wesley, Reading, MA (1975)

Appendix A Excerpt from the Final Coding of the Design Session

Time event begins	0–2	FBS	Strategy	Transcribed Speech
00:16:06			Ds	You bet. So I've been given the assignment and I'm reading the assignment I'll read it out loud I guess in part
00:16:06	0	R F	Cp	(reads brief) That's not a Batavus Buster over there..but..OK
00:18:00			X	*X No that one.. I could have.. I was going to say this at the end of your reading the assignment, that is not the Batavus Buster, that is a typical bike, but that is the backpack which is referred to.*
00:18:10	0	R F	Cp	OK. (continues reading brief aloud)
00:19:14			Ds	OK so I am going to... make a concept design for the device and this is a ...I will write this down so that I will not forget it
00:19:27	0	R S	Ps	..carring fastening....device.. alright.. and it is to attach to a bicycle, a mountain bike and to me that makes it different. Em, mountain bike....
00:19:51	0	R B	Ap	and this happens to be a ..em..
00:20:01	0	R B	Cp	OK let me just verify that this a..the um.. em alright and um..alright so it's a.. the HiStar backpack is a framed backpack it's an external framepack external.....
00:20:24	0	R B	Ap	OK so we know that it's actually an existing frame backpack it's an existing bike
00:20:35	0	R S	Ps	and we're making a device that is going to attach one to the other.
00:20:40	0	R F	Ap	Em and I have to focus on ease of use...good looks.....technical issues....price..
00:21:01			Ds	OK...Em OK em and the first thing I would do as a quick way of getting started is to ask you for the picture of a prototype and the test report.
00:21:27	0	R F	Cp	Em and er and you have the preliminary design for the device em but anyhow that'll show me that in this em..
00:21:37			X	*X So you would like the preliminary design, the design of the prototype which has been...*
00:21:42			Ds	Yeah right right there's no sense in designing something..er..if it's already.. there's no sense in starting from scratch if you can start at square two instead ofsquare one or square zero and um I (inaudible)
00:21:54			X	*X These are the sheets*
00:21:56	0	R F	Cp	get some ideas of what's not acceptable ha ha
00:22:00	0	S	Co	Alright and er these we know are um designs that the chap had worked out OK.
00:22:07			X	*X You also asked for a report. Which report was that?*
00:22:10	0	B	Cp	Em it says something about the em test report,

Time event begins	0–2	FBS	Strategy	Transcribed Speech
00:22:14	1	B	Ap	OK and it probably right off the bat says the backpack's too high or something like that and that bicycle stability's an issue . .
00:22:22	1	S	Ps	em off bat immediately my impression is hey it's nice to have putting it on the front handlebars
00:22:29	1	B	Ju	because you uh like low inertia on the handlebars
00:22:34			X	*X This is the er User Trials report.*[Experimenter adjusts window blind]
00:22:36	0	B	Cp	OK Look through this thing here it's em OK so it is a.....OK let's just see so I've got an idea of what this backpack design looks like em eh
00:23:05	1	B	Ap	interestingly enough em..... let me see... I see how the backpackinterfaces interestingly enough em... doesn't directly take advantage of the eh frame .. nor the fact that there is a frame. The prototype interfaces (inaudible)
00:23:40			X	*X One thing to say Dan, I'm sorry to interrupt you but if these sheets are, we'd like you not to mark on them but if you want to draw on them we can get you photocopies made straight away.*
00:23:48	0	B	Cp	OK no that's OK. Em most important conclusion to our (reads aloud from report) OK (continues reading aloud) OK
00:24:20	0	F	Cp	The product was considered ugly .. alright Takes a while to get used to cycling with this weight..
00:24:29	0	B	Cp	OK now em the ugliness is related to the frame itself OK Takes a while to get used to the way the attachment device itself because thats what the device is is.
00:24:46	1	B	Cp	Er the weight of the frame of the fastening device was er enough to cause problems,
00:24:56	1	B	Ap	well that's interesting it means that it was made too heavy probably
00:25:00	1	B	Cp	and no location on the centre of gravity attaching the fastening device to the pack tended to put the rear fixing points too low

12 Comparing Paradigms for Describing Design Activity

Kees Dorst and **Judith Dijkhuis**
Delft University of Technology, The Netherlands

This collection of papers will help to show that there are many ways of describing design processes. Each researcher will have attacked the design process in his or her own way, based on a unique choice of assumptions and goals. In this paper, two basic and fundamentally different ways of approaching the design process will be discussed, and evaluated on their descriptive value.

1 Two Paradigms for Describing Design Activity

Over the years, many systems for describing design processes have been developed. The 'first generation' methods of design methodology in the early 1960s were heavily influenced by the theories of technical systems. The positivist background of these theories made for design being seen as a rational (or rationalizable) process. Criticism of these models raised interest in the fundamentals of design theory, the logical form and status of design. It also fostered a need for more detailed descriptions of the design activity, leading to more attention for designers and design problems, rather than just for the design process.

Problem solving theories introduced by Simon[1] provided a framework for this extension in the scope of design studies by allowing the study of designers and design problems within the paradigm of technical rationality. Simon also provided a sound, rigorous basis for much of the existing knowledge in design methodology. This paradigm, in which design is seen as a rational problem solving process, has been the dominant influence shaping prescripitve and descriptive design methodology