DIVISION OF COMPUTER SCIENCE

The Role of Graphical Representations in Early Design and Requirements Specification

(Presented at The 16th International Conference on Software Engineering, Sorrento, Italy, 16-20 May 1994)

John Sapsford-Francis
Sara Jones

Technical Report No.196

May 1994

Technical Report No. 196 May 1994

The Role of Graphical Representations in Early Design and Requirements Specification

John Sapsford-Francis, Sara Jones

J.Sapsford-Francis, S.Jones@herts.ac.uk
School of Information Sciences, University of Hertfordshire,
College Lane, Hatfield, Herts, AL10 9AB, UK

Tel: (0707) 284354, 284327
Fax: (0707) 284303

The Role of Graphical Representations in Early Design and Requirements Specification

John Sapsford-Francis, Sara Jones

J.Sapsford-Francis, S.Jones@herts.ac.uk
School of Information Sciences, University of Hertfordshire,
College Lane, Hatfield, Herts, AL10 9AB, UK

1. Introduction

This report describes some initial investigations into the role of diagrams in early system design and specification. This early design and specification work is of critical importance to the success of the final product yet it is fraught with difficulty. Decisions made in the design stage have far reaching, and sometimes devastating effects on subsequent development work. Adopting a user-centred approach to system development adds a further layer of complexity which must, in turn, be addressed. There is considerable experience at the University of Hertfordshire (UH) of developing novel human-computer systems through user-centred design (see, for example, Bearne et al 1994). This experience provides a strong focus for our interests in enhancing the quality of the development process by integrating formal methods with the less formal techniques currently used.

We are particularly interested in design contexts where:

- highly interactive software is being designed and developed
- novel solutions to real world problems are required
- design teams are made up of individuals with varying expertise
- there are no fixed roles within the design team and membership may vary over time

From our initial investigations and from our own experience as designers it is clear that design under such circumstances is a complex and difficult process. It is often a problem-solving activity which relies on knowledge based performance (Rasmussen, 1974). Designers use this kind of problem-solving approach to tackle novel problems through the use of analogical and heuristic reasoning and the application of generic problem solving approaches. They typically use a wide range of informal representations. In design meetings associated with ongoing projects at Hertfordshire, members of design teams often use diagrams of various forms to communicate their ideas and form a basis for discussion. There is clearly much to be learned from experience in these areas about the process of design, about the role of diagrams within it, and about what aspects of the process - and the corresponding use of diagrams - may be improved.

A range of different activities are involved at different stages of the design process. Early on in design, there is a need for learning and exploration of the problem domain, a search for creative solutions, discussion, communication, decision making and agreement. As design progresses, there is also an increasing need for precision and reduction in ambiguity.

The diagrams commonly used by designers at UH to help them to reason about particular system designs are often unstructured and do not use any one set of drawing conventions. The level of detail often varies in different parts of a diagram, depending on the difficulty (to the designers) of describing the corresponding parts of the system. Designers also introduce new diagrammatic conventions to describe problems encountered for the first time in new domains or with novel system architectures.

In a close-knit design team, the fluidity of diagram semantics may not cause a problem, with the designers communicating effectively, even through changing and informal specifications. However, in larger design teams whose members have various levels of experience, the studies described below have shown that

misunderstandings can easily arise. The backtracking from formal specifications or prototypes developed by individual members of the design team on the basis of such mistaken understandings is at best time-wasting, and can have more serious implications if customers have also been misled by the informal descriptions.

As design issues become better understood, there is a need for notations which allow requirements and proposed solutions to be specified with greater precision, and which therefore support more effective communication between designers and users. Software engineers are looking increasingly to the use of formal methods and notations to provide such precision. Studies have shown that using formal methods early in system design tends to reduce the overall cost and development effort for a project by forcing designers to confront inconsistencies and possible problems with a proposed design at an early stage (Hall, 1990). In recent years, there has been increasing interest in the application of such methods in the process of human-computer interface design. At the University of Hertfordshire, we have begun to investigate the use of state-based formal notations such as Z to specify properties of graphical interface components (Jones, 1993), and process-based formalisms such as CSP to describe dialogues with hypermedia information systems (Jacob and Jones, 1994). We aim in future to investigate the role of proof in the formal specification of elements of the human-computer interface and in verification of user requirements.

We increasingly believe that there may be something to be gained from having, towards the end of the design process, a reasonably formal model of what is to be produced in design contexts such as those identified above. Increasing the formalism of the design solution as it is developed reduces the scope for mis-understanding between the designers, and facilitates identification of incompleteness, inconsistency or infelicity in the model. For larger systems, it may also be helpful to have a formal model from which important properties of the system to be built - perhaps relating to usability - can be proven to hold. Where large or important systems have been commissioned by particular bodies, it may also be possible to use a formal specification as the baseline for a contractual agreement about what is to be provided.

A focus of interest for us now is the way in which informal diagrammatic descriptions of various aspects of a system design might be linked to more formal specifications of those properties in a way which will permit interface designers to derive benefits of formal methods such as those envisaged by software engineers. Managing the transition from an incomplete, imprecise, ambiguous, and probably inconsistent initial understanding of the design problem to reasonably formal model or specification of what is to be built is a difficult process. Yet it must be done. Even if no formal model of the required system is built it is still necessary to construct programs which are expressed in the formal system of the programming language. We believe that diagrams and graphical representations may play an important role in mediating the process of increasing formality through the design phase, and have begun to investigate how this may be achieved.

2. Background Theory

2.1 Design problem solving

Rasmussen (1974) distinguishes between knowledge based and rule based performance in problem solving:

- **knowledge based** problem solving is required when tackling novel problems. It may involve the use of analogical and heuristic reasoning and the application of generic problem solving approaches.

- **rule based** problem solving occurs when tackling familiar problems, where the problem solver has built up a body of expertise. This kind of knowledge is called "rule based" because some theorists believe it can be expressed, adequately, in terms of production rules.

We argue that initial systems design requires a combination of knowledge-based and rule based performance In knowledge based performance designers typically use a wide range of informal representations in their search for novel solutions. This emphasis on novelty, suggests that creativity has much to do with

knowledge based problem solving (the role of creativity in design is discussed in the next section). Rule based performance is likely to require problem representations that have greater consistency and precision as designers seek to correctly recognise problem scenarios and apply appropriate rules to them.

One issue in bridging the formality gap is achieving sufficient support for both kinds of problem solving and enabling an effective transition from informal to formal. We believe that one solution lies in the use of representations: informal, semi-formal and formal.

2.2 The role of creativity in design

Creativity appears to be an essential part of software design problem solving.

Product design necessarily requires the search for novel solutions. For example in the development of the SPIRE system (Bearne et al 1994), one of the projects described in this document, the objective was to develop a multimedia system that would help both staff and students with the integration of students with disability into the Higher Education environment. In developing this system, much effort was expended on trying to devise innovative yet appropriate ways of providing access to useful information.

Furthermore the current development of information systems has frequently been criticised as being reactive and failing to meet the promise of innovative solutions to the needs of business. (Avgerou, C. and Cornford, T. 1993)

If creativity is an important part of design the various tools and techniques provided should be such that, at very least, they allow it to take place. To be able to judge the appropriateness of existing tools and techniques it is important to understand the process of creativity. What does creative problem solving behaviour involve? What support, as a design activity, might it require?

One widely adopted means of encouraging creativity is brainstorming (Osborn A.F. 1953). Although there is some doubt about the efficacy of brainstorming in groups is accepted as a generally effective approach for the generation of a large number of good ideas (Meadow, A. Parnes, SJ. Reese, H 1959). Brainstorming involves the generation of many unusual ideas whilst avoiding the premature evaluation of those ideas. This suggests that creativity at least requires the fluent generation of ideas and a postponement of evaluative judgement.

This suggestion is supported by Getzels J. and Csikszentmihalyi M. (1976) study of creativity in artists. These authors identified an important pattern of behaviour in successful artists: Problem finding. This behaviour is typified by:

- spending more time exploring approaches to work before settling on one
- remaining ready to change directions when new approaches suggest themselves
- not viewing a work as fixed even when it is finished

This suggests that representations for creativity should maximise fluency, discourage premature evaluation, encourage explorations, allow changes in direction etc.

The use of appropriate conceptual inventions is an important part of problem solving (Bransford et al. 1987). Intuitively one can see that having adequate concepts for description and reasoning is important. In novel problem solving situations it may be desirable to be able to introduce novel concepts. Furthermore concepts should be at an appropriate level for the task. For example it is widely accepted that a programming language provides too detailed and specific a representation for high level design reasoning. Working at an appropriate level of abstraction allows the construction of more generic designs that are more easily maintained.

Representations that support appropriate abstractions may also be important for analogical problem solving.

Gick and Holyoak (1983), for example, make that point that analogical transfer is facilitated by appropriate abstraction. However it is not easy to say what kind of abstraction is likely to provide the most fruitful analogical mappings. Perhaps all that can be said is that various abstractions should be considered.

There are further indications that the nature of a problem solver's representation of a problem influences how creatively problems are solved. For example: functional fixedness occurs when problem solvers have introduced too many constraints, perhaps unconsciously. Looking for alternative ways of representing problems helps. Glucksberg, S. & Danks J.H. (1968) for example found that labelling of objects when solving problems does not always help, sometimes it hinders. Labelling an object as a screwdriver prevents us seeing it as a potential circuit connector. Using nonsense names apparently makes it significantly easier to see new uses for objects.

We might conclude then, that in creative problem solving, it is important to have adequate concepts for the problem at hand, but that it is important that the problem solver considers each concept in an open way; accepting that there may be many views of the use and significance of a particular concept, exploring a variety of abstractions. Adopting this approach, while appropriate for exploring creative approaches to solving problems in knowledge based problem solving (Rasmussen 1974), is potentially disastrous once there is a requirement for rule based problem solving.

Some approaches to creative problem solving (Rickards 1990) recommend that there should be an opening up phase, where ideas are considered flexibly and possibilities are explored, followed by a closing down phase, where ideas are selected critically and developed into useful solutions. In the studies reported in this paper we were particularly interested in the way that opening up and closing down problem solving activities occurred and how various representations were used in relation to them.

2.3 The role of representations in design

One useful framework for thinking about the use of representations in design problem solving is Norman's (1991) action cycle (see figure 1). The action cycle is made up of two parts: the **execution** of actions to meet goals and the **evaluation** of the effects of these actions on the environment. "The **gulf of execution** refers to the difficulty of acting on the world.... The **gulf of evaluation** refers to the difficulty of assessing the state of the environment." (Norman 1991)

Where these difficulties are extreme they may be bridged through the use of representational objects . It is unsurprising, given the complex nature of design problem solving, that a wide variety of representations (for example: formal specifications written in notations such as Z and CSP dialogue specifications; semi-formal representations including data flow diagrams, state transition diagrams and entity relationship diagrams) are prevalent in software engineering.

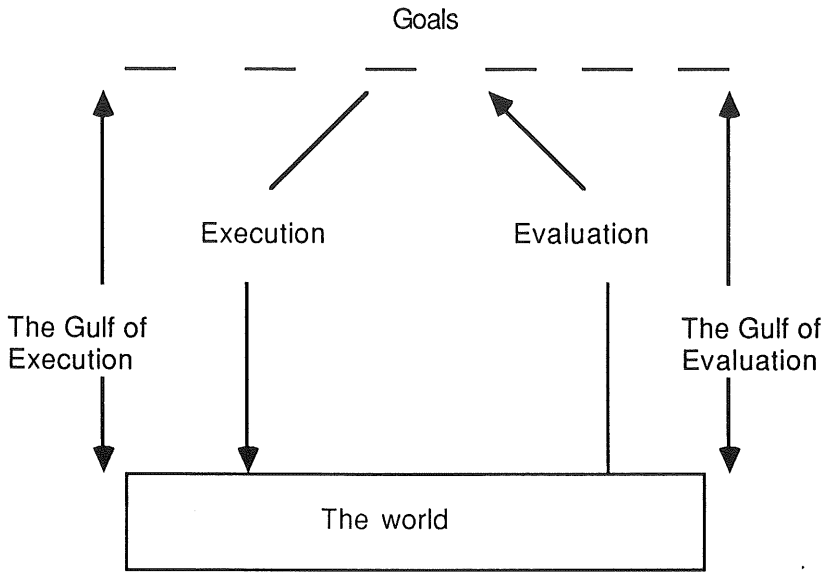


Figure 1: The action cycle; after Norman (1991)

Where, when using a particular representation, the designer finds the tasks of execution and evaluation too difficult it is often the case that further representations are introduced. This state of affairs is illustrated in figure 2 below: if it is difficult to construct a primary representation (e.g. a data flow diagram) the designer may resort to some secondary representation (e.g. an informal diagram) to help with its construction.

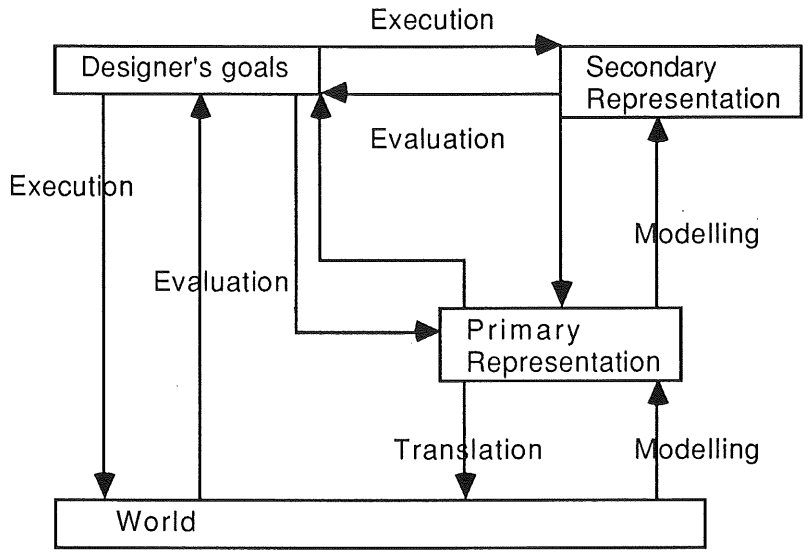


Figure 2: The role of primary and secondary representations in design

In the development of software it is considered good practice to employ a secondary representation for the purpose of design (and explanation of design to others). The primary representation, the software itself is considered too complex and too detailed to be effective for design. Traditionally software design has been carried out through the medium of various semi-formal notations (for example the structured notations: data flow diagrams, entity relationship diagrams, data dictionaries). In the last decade a strong case has been made for the use of formal languages as design and specification media. It is quite possible that semi-formal representations may be used as an aid in the construction of formal reprsentations. These formal representations can then be translated into software. Using the overview shown in figure 2 we can view this

use of a semi-formal representation as the introduction of a tertiary representation.

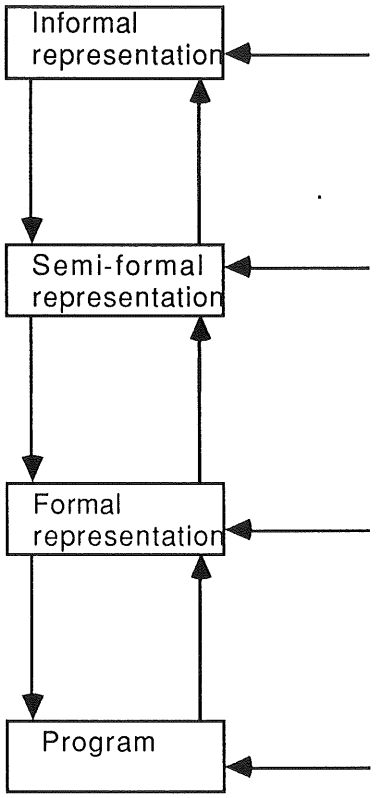


Figure 3: Relations between representations of varying formality

The creation of semi-formal diagrams is itself not without difficulty and it is quite common for designers to employ informal diagrammatic representations prior to producing any of the other representations. Thus a quaternary representation may be employed. The state of affairs shown in figure 3 is one possible way that these various representations may be used. However it is quite possible and often the case that any of the representations: informal, semi-formal and formal, is used to help understand and create any of the others. Designers may employ any combination of these representations with the program representation to help them understand and design.

3. Initial Questions About Diagrams in Design

The discussion in the previous section has lead us to identify the following initial questions as worthy of study:

- Why are diagrams produced?
- What makes a good informal diagram for the purposes of knowledge based problem solving?
- What kinds of diagram best support communication between designers in a meeting and agreement or consensus?
- What characteristics of a diagram will make it suitable as a basis for the derivation of a formal specification?
- How can we derive more formal representations and specifications from diagrams drawn during design?

These questions will be discussed in the remainder of this document.

4. Initial Investigations

This section describes two informal studies conducted at the University of Hertfordshire with a view to gathering information relevant to the questions above. These two studies were conducted in the context of an ongoing project at UH concerned with building an advice and information system to support the integration of students with disabilities into higher education. This system, SPIRE, uses multimedia technology and is intended to be highly interactive. It is intended to provide a novel solution to the real problems of those working to support students with disabilities in higher education. The team of designers involved in its development includes 5 individuals, though not all of these people are always present at meetings in which the design is discussed. The members of the team have varying levels of experience: three have worked together on projects in the past, but two are new to UH and do not share this experience.

This setting is therefore a good example of the kind of design context described in section 1.

4.1 Study 1: The Use of Informal Diagrammatic Notations in Design Meetings

This study was based around a fairly difficult meeting of the SPIRE project design team at which 4 members of the team (J, S, M and F) were present. During the meeting the designers were trying to deal with a number of important but intangible design decisions to do with the conceptual design of the system. A number of diagrams were drawn.

4.1.1 Aims

The aims of this study were to investigate the role of informal diagrammatic representations in design meetings. The following questions were of particular interest:

1. Why are diagrams produced?
2. What makes a good informal diagram for the purposes of knowledge based problem solving?
3. What kinds of diagram best support communication between designers in a meeting and agreement or consensus?

4.1.2 Procedure

After the meeting the diagrams were collected and the participants were interviewed. They answered questions about each of the diagrams. The diagrams were photocopied, although the originals were available to be viewed. Each participant was asked to:

1. Say who drew each diagram
2. Put the diagrams in order of their creation
3. Say what issue surrounded the creation of this diagram ("Why was this diagram drawn? What main issue does it relate to?")
4. Label the important parts of the diagram on the photocopy

Four diagrams were produced. These are referred to as: the 'layer' diagram, the 'local' diagram, the 'triangle' diagram, and the 'classes' diagram. Two of the diagrams (the 'layer' and the 'local' diagram - see figure 4) will be discussed in detail.

4.1.3 Results

There was broad agreement about who drew each of the diagrams and the order in which they were produced. A large number of statements were collected about the issues that surrounded the creation of the diagrams. For **some** of the diagrams there was considerable agreement amongst designers (including the creators of the diagram) about the issues they related to, but for other diagrams there was much less agreement.

Figure 4a: The Layer Diagram
(Showing labels 1 - 9)

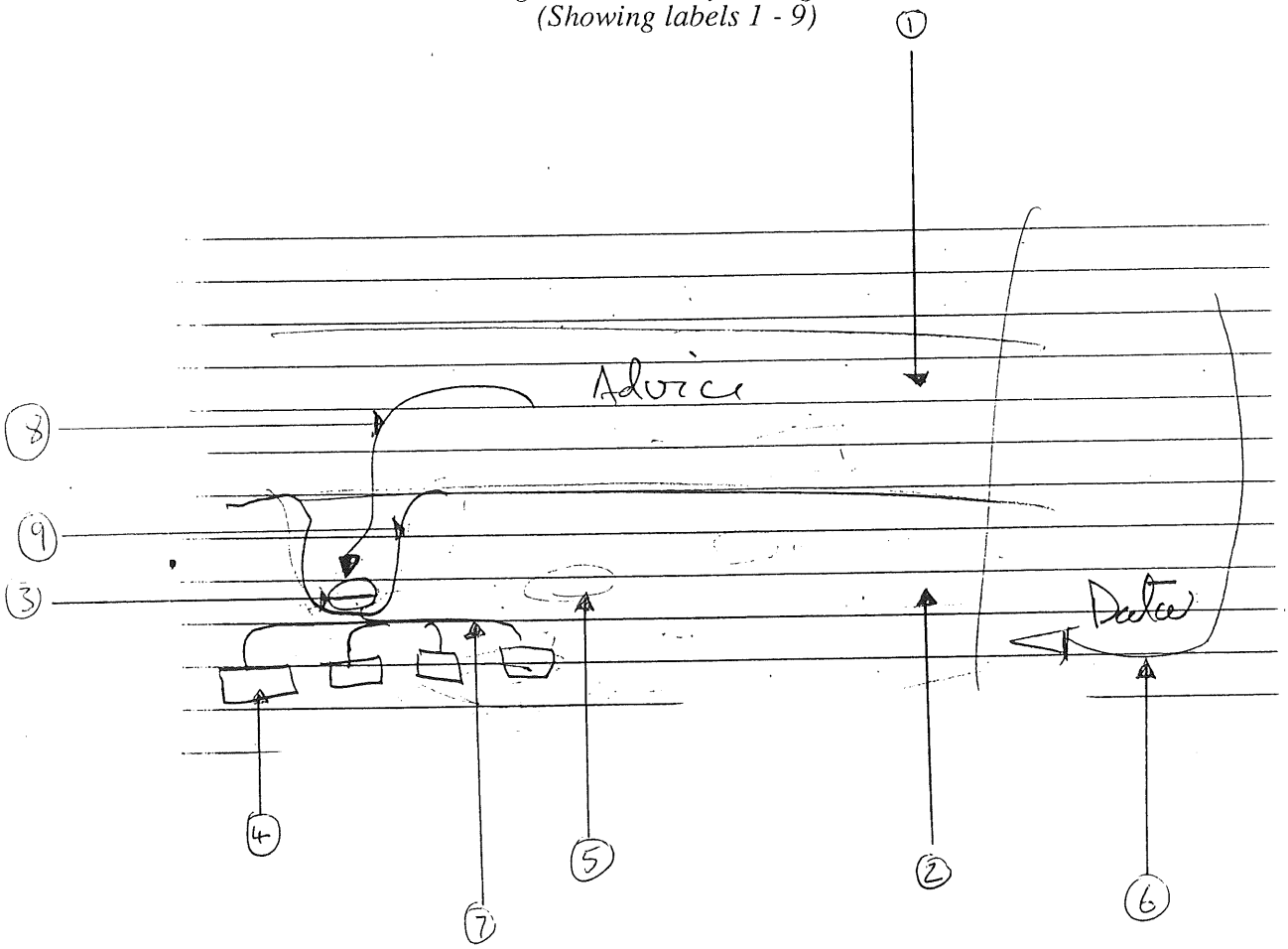
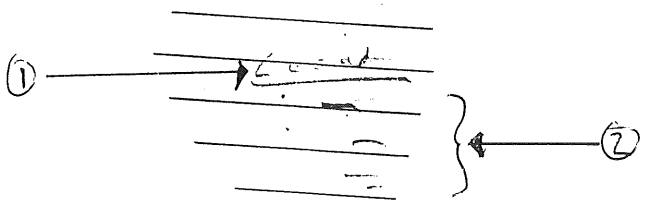


Figure 4b: The Local Diagram
(Showing labels 1 and 2)



a) The 'layer' diagram

All four designers said that the layer diagram was constructed to help decide where class descriptions of data objects should be presented to the user: whether they should be placed in the advice layer -essentially a hypermedium - or in the data layer -a structured database that the user can search or browse. The designer's responses to question 3 are given below:

J: "We were discussing whether class descriptions should be part of advice or part of the database"

S: "Having problems to do with what is going in advice layer and data layer. Particularly where class descriptions should go"

M: "Talking about class descriptions and where they fit into our three level hierarchy. There was some argument about whether it went in the bottom layer or the middle. User's and designer's models: F said does it matter form the user's point of view? J argued for the bottom layer, F argued for the middle layer"

F: "I drew this after the other triangular diagram. Its about the three layer model. Whether classes of data should be in the advice layer or the data layer. We decided this confused user and designer views"

This agreement between designers over the issues surrounding the layer diagram suggests that informal diagrams can be an effective aid to communication in contexts such as these.

Broad agreement is also shown in the labelling task carried out in relation to the layer diagram as shown in the table below.

	J	S	M	F
Advice Layer	•	•	•	
Data Layer	•	•	•	
Class Description	•	•	•	•
Entities / Data Organisation	•	•	•	•
Other Class	•			
User Access	•	•	•	•
Data / Class Link		•		
Advice Class Link		•		•

Table 1: Features of the 'layer' diagram labelled by each subject

b) The 'local' diagram

The observations described above contrast with those relating to the local diagram (see figure 4). Here it was hard to identify any one shared view. The designers' statements about the issues to which the local diagram related are given below:

J: "J describing contents list like a help system. Subtopics for local information. We discarded this...or did we?"

S: "J responding to F's prompting about class description "local". She was proposing that people should access the data via a hierarchical contents list."

M: "To do with things having lots of parents. Instances relating to multiple class descriptors. Something to do with an implementation issue to do with whether we should keep lists of classes attached to objects."

F: "J drew this to show that we could just have a list of local equipment rather than links. I felt we were assuming that data was just equipment at this point."

Labelling of the features in the 'local' diagram was also inconsistent as shown below:

	Feature 1	**Feature 2**
J	topic heading	sub topics
S	class description (?)	class descriptions indentations show 'is a subclass' relationships
M	local class	lot of local objects/equipment
F		class members

Table 2: Labels given to features of the 'local' diagram by each subject

4.1.4 Some Answers

Why are diagrams produced?

From the diagram creators' statements, it is apparent that informal diagrams of the kind shown in figure 4 may be produced for at least the following reasons:

- to show distinctions between similar system features
- to remind designers of previous decisions:

  The 'triangle' diagram was said to be drawn: "To show the distinction between links to the database and links to the advice and to remind everyone about the "triangle". The issues was what the name for a piece of advice represented."

- to help decide the place of proposed objects in the overall design:

  The 'layer' diagram: "I drew this after the other triangular diagram. Its about the three layer model. Whether classes of data should be in the advice layer or the data layer. We decided this confused user and designer views"

- to help discuss category membership relationships:

  The 'classes' diagram: "Discussing whether an entity belonged to one class or several. Mainly talking about bits of equipment."

- to help describe proposed system features:

  The 'local' diagram: "J describing contents list like a help system. Subtopics for local information. We discarded this...or did we?"

- to explore possible design decisions

The remaining questions (What makes a good informal diagram for the purposes of knowledge based problem solving? What kinds of diagram best support communication between designers in a meeting and agreement or consensus?) will be explored after discussion of study 2 in the next section.

4.2 Study 2: Semi-Formal Notations for Recording Design Decisions

This study was done following a further SPIRE project meeting at which a high level design for part of the SPIRE system dialogue was discussed. Four designers were present at the meeting (S, M, P and J). At the end of the meeting, each of the designers' own graphical representations of the dialogue agreed upon were collected. These had been drawn during and after discussions in the meeting which related to a fifth drawing on the white board. Unfortunately, this diagram was erased by the cleaners before it could be copied into the record of the study!

4.2.1 Aims

The aim of this study was to investigate the use of various diagrammatic or graphical representations in high level dialogue design. In particular to:

1. see what different designers in the meeting thought was important in the design discussions and decisions we made by asking them what the diagrams they drew in the meeting were about and looking at the way they labelled them (see steps 1 & 2 of the procedure below)

2. see if each designer's memory for the design decision reached was the same by comparing the decisions recorded in each of the notations (see step 3), and to see whether agreement was apparently greater when decisions were recorded using one notation rather than another

3. see whether any of the chosen notations seemed more or less appropriate than the others for recording decisions about dialogue, by considering:

- the level of agreement in diagrams drawn in the same notations by different designers
- the time taken to draw diagrams
- comments made by the designers about their use of the different notations during the study

4.2.2 Procedure

In this study, each participant was asked to:

1. Describe in general terms what your own diagram is about.
2. Label all important parts of your own diagram.
3. Draw a representation of what you think was decided upon in the meeting using:

- a flowchart
- a state transition network
- a JSD diagram

using the prompt sheets provided to find out what symbols to use for each notation, and what those symbols should mean. Drawings could be done in any order.

4. After each drawing, describe any problems with using any of the notations: e.g. things you wanted to represent but couldn't, difficulties with understanding how to use the notations, problems with interpreting your diagrams.

5. Note any further comments e.g. about how the notations compare.

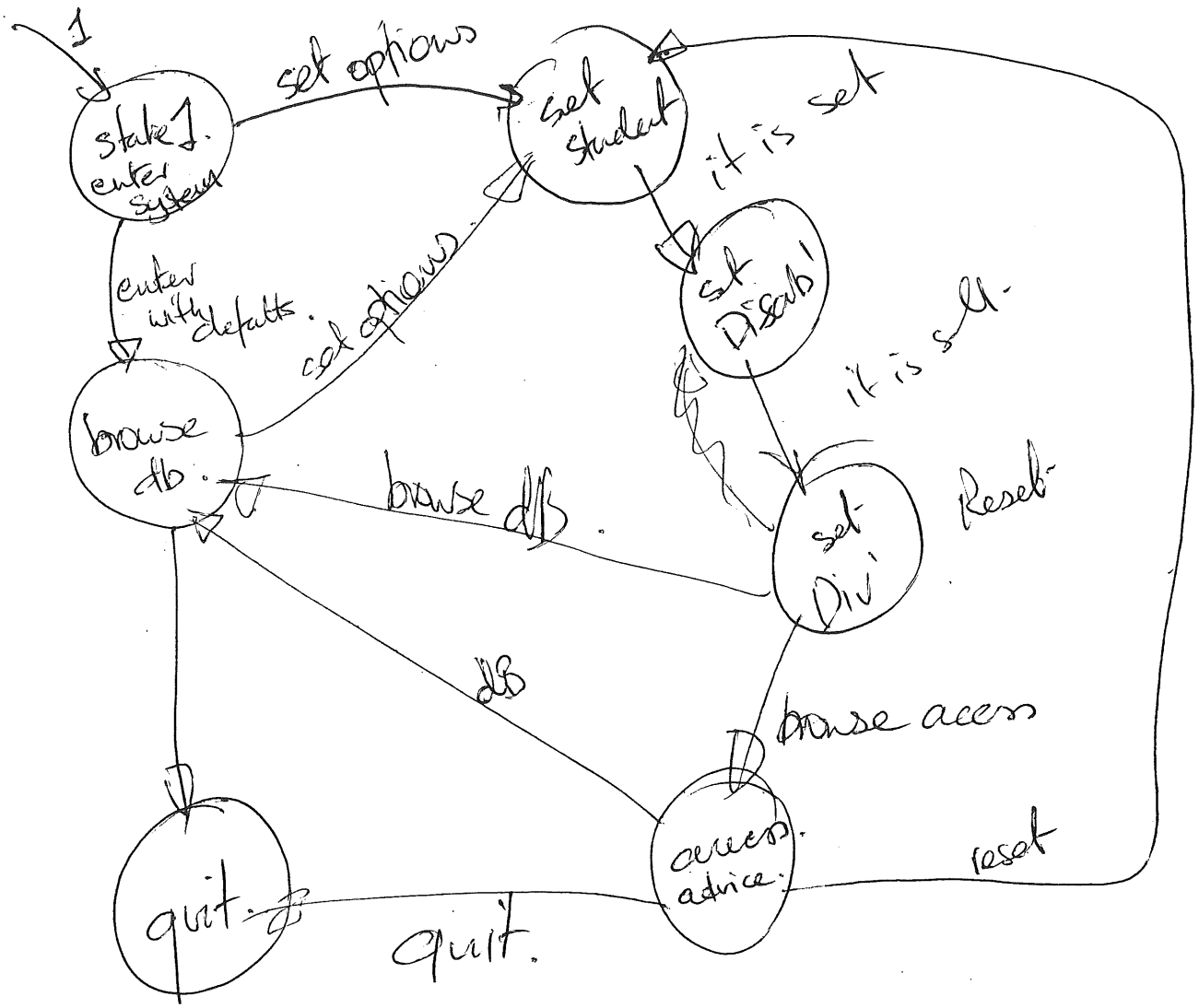


Figure 5: STN produced by designer P for study 2

4.2.3 Results

There was broad agreement between designers as to what issues had lead to their own diagrams being drawn. However, while three designers (S, M and J) had drawn their diagrams to record the design decisions reached during the meeting, one (P) said: 'My diagram was meant to be used as a discussion tool, not to record the design decision taken'.

The same kinds of elements were labelled in each of the designers' own diagrams so it appeared that a reasonable consensus regarding relevant design decisions had been reached during the meeting.

No consistent patterns were observable in the orders in which designers had chosen to tackle drawing diagrams using the three notations requested, or in the times they took to produce those diagrams.

The characteristics of diagrams drawn using each of the notations listed above were as follows.

a) Flowcharts

Early analysis of the diagrams in terms of the occurrence of symbols given to the participants on the prompt sheets shows that they were used in the following way.

	Total Processes	Total Conditions	Total Arrows
S	7	2	13
M	7	6	18
P	7	5	15
J	8	2	10

Table 3: Numbers of processes, conditions and arrows drawn by each designer.

Some processes (such as an initial Start or Enter process, and database and advice browsing processes) were identified by all designers. Others (such as Quit, Set Profile, and Browse Tasks) were only identified by some. The setting of user profiles was treated differently by different designers.

	S	M	P	J
Start / Enter	•	•	•	•
Quit	•	•	•	
View Subject Index	•			
Access / Browse Database	•	•	•	•
Browse / Read Advice	•	•	•	•
Set Profile	•	•		•
Select User Type, Disability and Division				•
Set Default Profile	•	•		•
Browse Tasks		•		•
Refined View of Database				•

Table 4: Processes drawn by each designer

The comments recorded by the designers after producing their flowcharts were as follows:

- Couldn't show details of 'setting profile' procedure. (S)
- Couldn't show that advice given depends on profile. (S)
- Ended up filling in bits of design I didn't feel were decided on in the meeting. (S)
- Not sure if I'm using flowchart syntax properly (horizontal arrows). (S)
- Hard to fit ideas into this sort of framework. (M)
- OK for this decision as it lent itself to simple Y/N questions. (J)

- But I wouldn't use it for preference. (J)
- The flowchart was OK but suffers from the same drawbacks as JSD [see below] as well as being too procedural. (M)

b) State Transition Networks

Early analysis of the STNs produced by the four designers reveals that the numbers of states specified by all designers was the same (though what these states were varied considerably - see below), but that differing numbers of transitions were recorded.

	Total States	Total Transitions
S	7	9
M	7	14
P	7	12
J	7	8

Table 5: Numbers of states and transitions drawn by each designer.

Again, some states (such as an initial Start or Enter state, and database and advice browsing states) were identified by all designers. Others (such as Quit, Access Mode Choice, Viewing the Subject Index, and Viewing Task Hierarchy) were only identified by some. The setting of user profiles was also treated differently by different designers: some gave a more detailed description of the states involved in this process than others.

	S	M	P	J
Start / Enter	•	•	•	•
Quit			•	
Access Mode Choice	•			
Viewing Subject Index	•			
Database Access / Browsing	•	•	•	•
Advice Browsing	•	•	•	•
Viewing Task Hierarchy	•	•		•
Profile Setting	•	•		•
User Profile Setting		•		•
Default Profile Setting		•		•
Set Student			•	
Set Disability			•	
Set Division			•	

Table 6: States drawn by each designer

The comments recorded by the designers after producing their STNs were as follows:

- Assume can quit system at any point. (S)
- States are always linked with what's on the screen. (S)
- Could put some information about transitions on arrows but not enough e.g. couldn't describe that advice displayed depends on profile set and task chosen very easily. (S)
- Filled in some bits not decided in meeting. (S)
- Not sure whether the bits circled in red are valid system states/transitions - possibly this is just one transition? (M)
- I keep wanting to put more interface detail in re: where do you go after Select Task or DB Access. This was not discussed at meeting and could be misleading if added. (J)
- Of the three notations, the STN was the most expressive for the ideas I wanted to capture. (M)

c) JSD Diagrams

The following numbers of procedures were used:

	Total Procedures
S	19
M	12
P	9
J	10

Table 7: Numbers of procedures drawn by each designer.

Some procedures (such as the overall Using / Accessing Information in SPIRE, Direct Database Access procedures) were again identified by all designers. Many more components were identified only by some designers. The setting of user profiles was again tackled differently by different designers.

	S	M	P	J
Using / Accessing Information in SPIRE	• b	• b	• b	• b
Start	•			
Quit	• *			
Choose Access Route	•	•		
Direct Database Access	• ° b	• ° b	• ° •	• °
Guided Information	• ° b	• ° b		• ° b
View Subject Index	•			
View Database	• * • *	•• °		
Set Profile	• b	• * b	• b	• b
Select Task		• °		• *
View Advice	• *	• °		• *
User Sets Profile	• ° b	• °	•	• ° b
System Sets Profile	• ° b	• °	•	• ° b
Choose Profile				•
Default Profile				•
Choose User Type	• *		•	
Choose Disability	• *		•	
Choose Division	• *		•	
Set User = Student	•			
Set Disability = All	•			
Set Division = All	•			
Browse Data		•		

••: procedure occurs twice in the diagram
b : procedure is broken down further
° : procedure is defined as optional
* : procedure is defined as iterative

Table 8: Procedures drawn by each designer

The comments recorded by the designers after producing their JSD diagrams were as follows:

- Can't show relation between two 'view info from db' boxes. (S)
- Can't show swapping between advice and data layer or subject index and data layer (but not sure this decided in the meeting anyway). (S)
- Can't show that advice given depends on profile. (S)
- Good to be able to break down functions. (S)

- Diagram got unwieldy. (S)
- Not sure if this is the right way to represent choices. (S)
- Doesn't seem very suitable for designing flexible dialogues. (S)
- In general I found it harder to do this than the other two methods. I think this was because in the meeting we talked about conceptual design rather than implementation and I have always seen JSD as more biased towards implementation issues. (M)
- Found this harder than flow chart. (J)
- I don't think it makes the selection decisions very obvious. (J)
- Wasn't sure where to put the detail of profile selection. (J)
- Numbering looks cumbersome. (J)
- How do I show select one or the other? (J)
- I found the JSD too low level for high level design ideas and too restrictive. (M)

4.2.4 Some Observations

Reasons for drawing informal diagrams identified in study 1 were:

- to show distinctions between similar system features
- to remind designers of previous decisions:
- to help decide the place of proposed objects in the overall design:
- to help discuss category membership relationships:
- to help describe proposed system features:
- to explore possible design decisions

Some further reasons for drawing informal diagrams during a design meeting which became apparent in this study were:

- to make a record for personal use
- to make an 'official' or agreed record of a design decision

Comments about use of the three notations can be broadly categorised into three main areas:

- difficulties in expressing design decisions in this representation
- representation encourages specification of more detailed design than has been decided
- concern over diagram complexity or attempts to keep representation simple

Other points which arose in considering the results of the study were as follows.

Designers who are not experienced or trained in the use of a semi-formal notation may use it incorrectly.

One of the participants in the study was confused over whether they had used the flowchart notation correctly, some designers did not label all transitions in the State Transition Networks, and three of the designers used the JSD diagram notation incorrectly, despite having access to a prompt sheet which gave a general description of how to use it.

Different designers may use semi-formal notations in different ways.

With flowcharts, some designers used a condition to denote a point where the system asks the user for input and then acts according to the input received (e.g. J's condition: 'Do you want to?'). Others used a condition to describe something which the system does internally without the user being aware of it (e.g. S's condition: 'Profile set?').

In the State Transition Networks, there were differences in emphasis about what different designers thought should be represented as states. S's states corresponded to what might be on the screen at a particular point in the interaction. M's and J's were more task-related. Furthermore, some people put in states what others put in transitions. For example, S used a state to represent point where users could choose between the two available access modes, whereas other designers simply specified two possible transitions from the previous state. Some people gave more detailed descriptions of transitions than others. An example from S's diagram is: 'Profile set: use user type to choose task hierarchy'. Examples from J's diagram are: 'Yes', 'No', 'Select DB', 'Select Guided Information'.

Three of the designers used the JSD diagram notation in much the same way. The diagram produced by the fourth was much sketchier.

Using different notations apparently encourages designers to specify different things.

One example of this is the specification of a quit option. The possibility of quitting from the program under design was not actually discussed during the meeting, and no explicit decisions regarding quitting had been made in any previous meeting. Thus, the fact that J never specified how a user would quit from the program may reflect the fact that she was faithfully following the instructions of the study to specify only what she felt had been decided in the meeting. However, the other designers decided to incorporate a quit into their graphical specifications. Interestingly, each incorporated it (explicitly) in only some of their diagrams. While S included it explicitly in both the flowchart and JSD representations, she did not include it in the STN, but noted that she was assuming it would be possible to quit at any point. M included quit only in his flowchart representation, and P only in the flowchart and STN.

	Flowchart	STN	JSD
S	•	Assumed from anywhere	•
M	•		
P	•	•	
J			

Table 9: Occurrences of the Quit Option in Each Designer's Representations

It is too early to say what features of the notation are significant here, though we might speculate that the sequential view of interactions which flowcharts and JSD force on the designer are more likely to lead to the specification of a quit option as the final procedure in a chain of interactions. Specifying the fact that the user should be able to quit from the system at any time is never easy in a graphical notation and quickly leads to diagrams becoming too complex to be helpful.

Individual designers may record different design decisions using different notations.

When drawing the STN representation (which she did second), S stated that users should be able to quit from the system at any point. However, when using the other notations, she specified a system from which users could only quit after having done a number of other operations.

In her flowchart representation, J specified that users should be able to move from one part of the system (where they were viewing advice) to another (where they would be able to see a refined view of the database). In her STN, she did not specify that this should be possible.

Again, it is too early to say what features of the notations lead to these differences, though we suggest that the sequential view embodied in flowcharts and JSD diagrams is again significant.

Some differences in designers' views and recollections of design decisions remain constant when decisions are recorded in different notations.

Sometimes, designers put in things which were not discussed at the meeting of interest here but were assumed to follow from previous meetings. Not all designers agree on these issues. For example, S recorded the fact that users should access one part of the system (the database) in a particular way (via a subject index), but no other designer mentioned this. The use of a subject index to access the database had been discussed at a previous meeting but not explicitly decided upon. As a result of that meeting, S apparently assumed that it was to be used in the way specified, but other designers did not.

P also had recorded a different and more detailed view of a particular part of the interaction (that in which a user's 'profile' is set) than other designers in each of his diagrams. It is interesting to note that the diagram drawn by P during the meeting was the only one to have been drawn 'as a discussion tool' rather than as a record of the design decision reached, so that his memory for what was decided might not have been so precise as that of other designers who recorded the decision in the meeting.

If diagrams using particular notations become too complex, designers may forget to put some things in.

For example, S recorded the fact that users would see a particular part of the system (the task hierarchy) in her flowchart and STN diagrams, but not in JSD (which was drawn third). Her JSD diagram had already become quite complex by the time she came to specifying the relevant part of the system - it practically filled the page already - and she may have forgotten to specify the use of the task hierarchy because of the size and apparent complexity of the diagram.

5. Conclusions and Further Questions

Design problem solving can be usefully viewed as consisting of knowledge-based and rule-based problem solving (see the discussion of this in section 2).

Knowledge-based problem solving is employed when trying to deal with novel problems. It necessarily involves such approaches as analogical problem solving, heuristic reasoning and creativity. We argue that such problem solving behaviour is associated with the use of informal diagrams. Rule based problem solving is employed when solving familiar problems appropriate to a body of expertise. This problem solving behaviour is equivalent to the application of problem solving rules. We argue that this kind of problem solving is the sort that might generally take place when reasoning with a formal specification.

The management of the transition from knowledge based problem solving using informal diagrams, to rule based problem solving using formal specifications is therefore an important issue in system design. Semi-formal notations frequently act as intermediary representations in this transition.

What makes a good informal diagram for the purposes of knowledge based problem solving?

The studies described in this paper do not provide a sufficiently large database to support definitive answers to this question. Some indications can be derived from a comparison of the 'layer' diagram and the 'local' diagram (figure 4) in study 1. For the 'layer' diagram there was better agreement and better memory for the issues addressed and the decisions made by the designers. Recall of the issues and decisions for the 'local' diagram were sketchy at best. This difference in recall and agreement may have been caused by many situational factors, however the following factors may account for some of the differences:

- much more contextual information was provided in the 'layer' diagram, the structure of the whole system was sketched in
- although more contextual information was provided in the 'layer' diagram it was sketched in using much less detail than the part of the diagram which was the focus of the discussion.
- the 'layer' diagram made much greater use of visual coding

Good memory for design decisions is an important fundamental requirement in effective design, however it is not necessarily the most important requirement. In the discussion of creativity in design (section 2) we suggested that the following diagrammatic features were desirable.

- it should encourage appropriate abstractions and conceptual inventions
- it should be flexible; it should allow changes in direction
- it should allow fluency
- it should encourage exploration
- it should discourage premature evaluation

Some support for these recommendations can be found in study 2 (see section 4). In this study the designers were encouraged to express informal design decisions using such semi-formal representations as flow charts, JSD diagrams and state transition diagrams. The designers' comments whilst carrying out this exercise were captured, broadly their comments can be classified under three headings:

- difficulties in expressing design decisions in this representation
- representation encourages specification of more detailed design than has been decided
- concern over diagram complexity or attempts to keep representation simple

The designers' comments are consistent with the features suggested on the basis of our initial theoretical analysis and therefore lend them some support.

What kinds of diagram best support communication between designers in a meeting and agreement or consensus?

As we described in section 4, mis-understandings can easily arise if diagrams used in design meetings are vague and informal. The members of the design team are likely to mis-understand each other most are those who do not share a great deal of experience - for example new comers to the design team (who will often be the programmers responsible for fine tuning the design!) are likely to mis-understand something which is agreed between senior members of the team on the basis of shared experience from previous projects.

We speculate that diagrams which are rich and expressive as well as precise and unambiguous are likely to form a good basis for communication and permit a reliable formation of consensus during design meetings. But some ideas are conceptually simple, and therefore easy to communicate, yet tedious or difficult - especially for people who specialise in design rather than the use of formal notations - to specify formally. Imposing the use of formalism where it is not necessary or appropriate will inhibit both creativity and communication.

What characteristics of a diagram will make it suitable as a basis for the derivation of a formal specification?

Different kinds of diagrams may be appropriate for considering early functional design, dialogue and screen design at different levels of detail. The exact choice of a graphical notation to lead naturally on to the development of a formal specification will also depend on the formalism in question. However, considering the properties of formal specifications in general (independent of the notation in which they are written), we suggest that it may be easier to derive such specifications from diagrams which share some of their salient properties such as:

- lack of ambiguity
- adequate expressiveness in relation to the design issues under consideration
- consistency in use of alphabets - in this case of graphical symbols.

How can we derive more formal representations and specifications from diagrams drawn during design?

Having one person go away after each design meeting and try to record formally what was decided is dangerous! As demonstrated by the differences in diagrams drawn in study 2, that person's memory may not be the same as that of the others present (unless measures are taken to ensure that it probably will be - see below).

It may be helpful, where possible, to use a notation closely related to the formalism to be used in specification during design meetings where relevant issues are discussed. Otherwise, the use of a formal notation which imposes a framework or way of thinking about the problem which is different to that used during the meeting is likely to lead to the specification being different to what was intended in the meeting. Similarly we believe that it may be easier to translate between an informal graphical representation and a formalised graphical notation than between an informal diagram and a text-based formal notation.

Taking the above points into consideration, we might propose that a consensus regarding design decisions should be achieved during each meeting, through the medium of a formalised graphical representation. This representation could be used as a basis for discussion, negotiation, and specification of agreement during the meeting - effectively playing the role of a joint contract agreed by all designers present. We propose that diagrams drawn using formalised graphical representations should be composed by one or more of the designers present in a public way and taking account of the views of all designers present - conflicts being publicly resolved as necessary. We suggest that this might be done using physical or computer-based (CASE) tools which support the creation of appropriate diagrams - for example allowing designers to manipulate symbols used in notations appropriate for describing the relevant design decisions (see above). We aim to continue our investigations in this area in order to further investigate these issues.

References

Avgerou, C. and Cornford, T. 1993. A Review of the methodologies Movement. Journal of Information Technology (1993) 5, 277-286

Bearne, M., Hewitt, J., Jones, S. and Sapsford-Francis, J. An Integrated Approach to the Development of Advice Systems to Support Learning and Domain-Based Information Retrieval, University of Hertfordshire, School of Information Sciences, Technical Report No. 180, February 1994.

Bransford, J.D & Sherwood, R.D., Sturdevant, T. 1987. Teaching Thinking and Problem Solving. in Baron J.B. and Sternberg R.J. (Ed.s) Teaching thinking skills: Theory and Practice. Freeman 1987.

Getzels J. and Csikszentmihalyi M. (1976).The creative vision: A longitudinal study of problem finding in art. New York Wiley

Gick M.L. and Holyoak K.J. (1983) Schema Induction and Analogical Transfer, Cognitive Psychology, 1983. 15, 1-88

Glucksberg, S. & Danks 1968 Effects of discriminative labels and of nonsense labels upon availability of novel function. Journal of Verbal Learning and Verbal Behaviour, 7, 72-6.

Green, T.R.G. 1989 Cognitive Dimensions of Notations, HCI '89. People and Computers V. Sutcliffe, A. and Macauley, L. (Eds.)

A. Hall. Seven myths of formal methods, in IEEE Software , September 1990.

L. Jacob & S. Jones, Formal Dialogue Specification for Hypertext and Multimedia Systems , Technical Report No. 175, Computer Science Division, University of Hertfordshire, January 1994.

S. Jones. Three-Dimensional Interactive Connection Diagrams for Knowledge Engineering. PhD thesis, City University, 1993. Chapter 11.

Karat, J. and Bennet, J.L. (1991) Working Within the Design Process: Supporting Effective and Efficient Design. in Carroll, J.M. (Ed.) Designing Interaction: Psychology at the Human Computer Interface. Cambridge 1991.

Meadow, A. Parnes, SJ. Reese, H 1959. Influence of brainstorming instruction and problem sequence on a creative problem solving test. Journal of Applied Psychology 43. 413-416 1959

Norman, D.A. 1991, Cognitive Artefacts. in Carroll, J.M. (Ed.) Designing Interaction: Psychology at the Human Computer Interface. Cambridge 1991.

Osborn AF 1953 Applied Imagination. New York. Scribner

Rasmussen, J. and Jensen, A. 1974 Mental Procedures in Real Life Tasks: A case study of electronic troubleshooting. Ergonomics, 1974, 17, 293-307.

Rickards, T. (1990) Creativity and Problem Solving at Work. Gower 1990