A DATA CENTRED FRAMEWORK FOR USER-CENTRED DESIGN

David Benyon
Computing Department, Open University,
Milton Keynes, MK7 6AA, UK.
Tel: +44 (0) 1908 652679 e-mail: D.R.Benyon@ open.ac.uk

KEYWORDS
Data, task, design framework, conceptual design, physical design

ABSTRACT
The notion of a *user task* needs using with care and needs placing in the context of a more abstract representation of the human-computer system. A framework is developed which provides a structure within which both existing tasks and new tasks can be considered. The framework focuses on an information processing view of HCI and distinguishes conceptual from physical design. In doing so it seeks to complement other perspectives (such as ergonomic, sociological , organisational etc.) rather than to replace them. The data centred framework enables designers to place the idea of 'task' within a considered cognitive approach.

INTRODUCTION

The essential principles of user-centred design remain broadly in line with the original formulation provided by Gould and Lewis (1985). These principles clearly provide good advice, but a major problem facing the system designer is to know how the principles should be applied.

Whilst there is much good advice available in the HCI literature on aspects of design, there are few methods or approaches which guide the designer. Even the more prescriptive of methods (e.g. Browne, 1994) encourage designers to pick and choose techniques according to the project at hand.

Other approaches to user-centred systems design typically emphasise the concept of a user *task*. (e.g. Carroll (1990), Johnson (1992)). Hix and Hartson (1993) argue that 'Approaching user interface development...*from a user and task view*, should result in *higher usability*' (p. 7, authors italics).

The assumption underlying the emphasis put on 'task' appears to derive directly from Gould and Lewis's (1985) first principle - to make user issues central in the design process. But does attention to *user tasks* correspond to making *user issues* central in the design process? Moreover, these statements do not make clear whether attention to tasks means designing to support existing tasks or designing new tasks in a more user-centred manner.

What is required is a more abstract representation of the human-computer system; a framework within which both existing tasks and new tasks can be considered.

THE TROUBLE WITH 'TASK'

The concept of 'task' has dominated the ontology of HCI since the earliest days. Tasks are part of Long's (1989) conceptualisation of HCI along with humans, computers and effectiveness. Moran's (1981) command language grammar (CLG) includes a task level of description and many task analysis techniques have been developed (Diaper, 1989).

One would imagine that such a ubiquitous term would have a simple definition, but this is not the case. Some authors argue that a task is device independent (Bösser, 1987) and others that it is device dependent (Shepherd, 1989). More recently, problems with the notion of task have been identified by Draper (1993). He identifies several different meanings for the term 'task' and warns against instantiating current tasks in future systems. The debate in *Interacting with Computers* (Benyon, 1992a; Diaper and Addison, 1992; Benyon, 1992b) raises a number of issues. Long now prefers the term 'work' in place of task and in a similar vein, Preece et *al* (1994) argue that 'work' is more appropriate than 'task' because of the distributed nature of many real world work situations. Fischer's (1989) design environments are to support 'human problem-domain communication' (p. 62).

In order to clarify the appropriate role and position of 'task' within HCI, we can adopt the following definitions.

A **goal** is;

a state of a system which the human (or, more generally an agent where an agent is any autonomous, rational, creature, machine or system which formulates its own goals and seeks ways of satisfying those goals) wishes to achieve.

For example, the human wants to write a letter, to produce a balance sheet, to find what is on television, and so on. A goal must be described at a particular level of abstraction.

A goal is achieved using some instrument, method, agent, tool, technique (mental or physical), skill or, generally, some *device* which is able to change the system to the desired state. Typically a goal can be accomplished using a variety of devices. For example, writing a letter is a goal which can be accomplished using a device such as a typewriter, a word-processor or pen and paper. Calculating the result of 385 ÷ 11 can be accomplished by using a pocket calculator, doing some mental arithmetic or using long division.

Given that the person has formed a goal, the person selects a device which (s/he believes) will enable him or her to achieve that goal. It is only once a device has been selected, that the tasks necessary to accomplish the goal may be understood. The tasks are prescribed by the logical structure and functioning of the device, that is, by the way that it has been designed, or has evolved.

Thus, we can define a **task** as:

the activities required, used or believed to be necessary to achieve a goal using a particular device.

A task is a structured set of activities in which actions are undertaken in some sequence. Certain actions may be repeated during the execution of a task. Alternative actions may be available which will accomplish some part of the task. A task is an activity which may include the sequence, selection and repetition of actions.

At some point, the human physically interacts with a device by performing an action . For example, the person types a command on a keyboard, physically moves a pointing device or presses a button on a display panel. There are also mental actions (or operations in GOMS (Kieras and Polson, 1985) terminology) such as 'retrieve from memory', 'recognise', 'compare' etc.. The significant difference between tasks and actions is that actions do not include any sequencing, repetition or selection (the control structure). Actions are at the skill level (Rasmussen, 1986).

Thus we define an **action** as:

a task which involves no problem solving or control structure component.

This definition of 'action' is consistent with the concept of a 'simple task' (Card, et al., 1983), 'unit task' (Payne and Green, 1989) or 'operation' (Kieras and Polson, 1985). Clearly however these definitions are only useful given some declared level of abstraction and given the knowledge and skills of particular users. A touch typist may consider typing the letter 'e' as an action, whereas for a novice this would probably be a relatively complex task consisting of the sequence of actions 'search keyboard', 'locate letter 'e'', 'press key'. Even then 'search' and 'locate' may themselves be relatively complex. It is these problems which Draper (1983) highlights in his analysis - the concepts of goals, tasks and actions need using with care and require an explicit statement of the context of the analysis to be useful.

The analysis above is not just intended to clarify definitions. The fact that a task is dependent on the device used to accomplish a goal means that *any change in device will change the task* (even if it does not change the user's goal). Even such a trivial change of device such as replacing a corded by a cordless telephone changes the task required to accomplish the goal of making a phone call. The functionality of the human-telephone system has not changed, but the onus of making the initial connection between handset and the telephone network (i.e. to obtain a dialling tome) has shifted from the device to the human. Whereas with corded telephones, lifting the receiver automatically established the connection, with the cordless phone the user has to press a button to do this.

A FRAMEWORK FOR DESIGN

The three levels of description of HCI activity lead to a two layered approach to design. **Conceptual design** is concerned with the goal and task levels and the mappings between these levels. It focuses on *what* the system has to be like if it is to meet its declared purpose, but not on *how* the structure and functions are to be realised in a physically instantiated system. The process of conceptual design results in a conceptual model of the whole human-computer system.

Physical design concerns the design of the physical system - embedding the conceptual model of a system in a physical structure (i.e. designing the device) so that users can communicate with that system. This involves both the operational aspects - what the system will do - and the representational aspects - what the system looks (and sounds) like. The operational aspects include making decisions about dialogue structure, the use of keystrokes, mouse movements, button pushes, etc. and feedback. Details of exactly how to display information, such as where to position items on the screen, the form of icons, dialogue boxes and how to use different media is the concern of the representational aspects of design

Moving from the conceptual to the physical level requires the designer to decide who or what is going to undertake particular functions and how information is to be distributed through the system. This is the process of **task allocation.** By allocating certain (logical, or conceptual) functions to humans, to the computer or to a human-computer system, the designer *creates* tasks for the human. By allocating certain data structures to the system the designer *creates* a demand on the user's cognitive processing. Within this framework, user-centred design is not concerned with *understanding* human tasks, it is concerned with designing devices which *impose* tasks on humans. The purpose of user-centred design is to support the goals which people wish to achieve - it does this by allocating tasks to human and to computer, with user issues remaining central to this activity. Thus the conceptual/physical distinction above is reflected in analysis as well as design. The designer needs to understand, conceptually, what the human-computer system is to do and to design, physically, how the human-computer system can achieve this purpose.

This framework seeks to simplify our understanding of the information processing view of HCI and to focus attention on the processes of HCI design. The psychological basis of the framework assumes that users can know something at one level but not at another. For example, a user may have knowledge at the conceptual level, i.e. a clear goal (such as deleting a section of text) and may understand the structure of the appropriate task (such as selecting a piece of text and then removing it) but not know how to do it at the physical level (for example, which sequence of keys to press, i.e. how to map the task structure onto the action structure).

It is also important to recognise that distinguishing between conceptual and physical design does not provide a method for the design; only guidance about what the design should involve. Analysts and designers will iterate between these two levels of description and will fix on certain physical design decisions in order to understand the conceptual level better. This iteration will involve various kinds of evaluation with users to check that the design really does meet their needs. However, the advantage of designing at the conceptual level before details of the physical design are fixed is important as it helps to avoid the problem of 'design fixation' and maintains a wide design space in which alternatives can be considered for as long as possible.

This description of levels considers the conceptual and physical components from both designers' and users' perspectives. Designers often fall into the trap of developing a conceptual model of a system to which the user has to adapt rather than the other way around. By analysing a design problem from the two perspectives the differences between the two can be highlighted and subsequently resolved.

CONCEPTUAL MODELLING

In order to develop a human-computer system, the designer must construct a suitable representation - a conceptual model - of the system. The purpose of conceptual models is to allow analysts to reason about a problem and come up with design solutions in the abstract. The importance of conceptual modelling has long been recognised in information systems design (Benyon, 1990, Checkland, 1981) and more recently in HCI (Braudes, 1991) A good conceptual model is one which is appropriate to the situation at hand. It highlights significant information and suppresses unnecessary detail. Conceptual models may be used to aid analysis - constructing the model forces the analyst to consider what is significant - as a formal design technique, to communicate ideas or to test hypotheses.

Models can be made of many things and can be about different aspects of the situation. Within HCI, we may want to model the social, political or organisational aspects, using techniques such as advocated within cooperative design (Greenbaum and Kyng, 1991). We may wish to model the physical interaction using a technique such as UAN (Hix and Hartson, 1993) or using storyboards and scenarios (Preece, et al. 1994). The constructs which a model employs, the ability to manipulate those constructs and the constraints which can be expressed are essential to its efficacy. The notation employed by a model can be critical to its usability.

HCI focuses on the interaction between humans and computers, so a conceptual model of the human-computer system which is both an abstraction of humans and of the computer system would be desirable. Moreover, one important aspect of HCI is the exchange of information and meaning. Using data models - models of some aspect of a human-computer system made of data. - is particularly appropriate for human-computer systems since the model uses a construct which is applicable to both human information processing and to computer data processing.

Data models can represent the structure of data in the whole human-computer system by using the technique of entity-relationship modelling (Benyon, 1990). They can represent the processing of data by using the technique of dataflow diagramming (DeMarco, 1979). They can represent the structure-function relationships through 'behavioural models' such as transition networks and entity life histories (Benyon, 1990; DeMarco, 1979). They can represent the meaning of concepts by representing data definitions using a data dictionary (DeMarco, 1979). Data models support the framework outlined above because they can be used at both the conceptual and physical levels and can be used to assist with task allocation.

The ERMIA technique (Green and Benyon, 1995) applies this philosophy to an analysis of structure. This paper focuses on a complementary use of data models; to represent the information processing in human-computer systems.

CONCEPTUAL DESIGN

In order to illustrate the framework in action and the power of data models, let assume that an initial analysis has been undertaken and that the following requirements specification has been produced:

Eurobank PLC, a mythical international bank, is designing an automatic bureau de change machine that resembles the autoteller machines we see in our high streets. The machine, Eurochange, is intended initially for installation in airports and will allow travellers to obtain the main European currencies quickly without having to find the nearest bank.

Three main processes have been identified; 1. Validate User which checks that the user has a correct type of card and correct PIN (personal *identification number), 2. Enter Requirements which allows the user to specify the amount of foreign currency required and 3. Deliver Currency which updates the credit card with the amount withdrawn and physically delivers the currency.*

Dataflow diagrams have a number of desirable properties. One of these is that they can represent computer inputs and outputs and the data processed by humans using the common concept of a dataflow. Similarly both human and computer processes can be represented (as circles). Designers do not commit themselves to a particular implementation by representing the flow of data in the system.

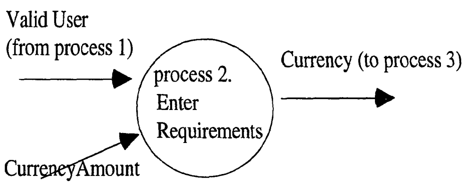

Figure 1 'Top level' dataflow diagram for process 2. Enter Requirements

The 'top level' dataflow diagram for process 2 is shown in Figure 1. It is a conceptual model of process 2. Notice that it says nothing about the flow of control (e.g. whether the data from process 1 - ValidUser - is processed before the CurrencyAmount is entered). It says nothing about how the data should be entered. It does not even say what the content of the dataflows are. All it says is that CurrencyAmount and ValidUser are two dataflows[1] which are processed by the Enter Requirements process to produce a dataflow called Currency. The contents of these dataflows have not yet been identified; conceptually process 2 requires CurrencyAmount and ValidUser as input and produces Currency as output. The abstraction achieved by the dataflow diagram is vital to good systems design; it hides the detail of the form and control structure of processes and dataflows.

One of the most important decisions to be taken during the development of a human-computer system is to allocate tasks; to human, to computer or to a human-computer system. The developer needs to establish who (or what) is going to provide the data or knowledge necessary to accomplish a task and who (or what) is going to physically accomplish the task. For example, in the Eurochange example it is difficult to imagine how the computer could provide the PIN. This must be a user action in order to meet security criteria. Similarly you would not expect the user to have to calculate exchange amounts. The computer should do this. Many other functions which are logically - or conceptually - required, however, can be allocated to the human, to the machine or to some combination of the two.

TASK ALLOCATION

Dataflow diagrams can be used to assist in task allocation; since the whole human-computer system is represented, options for distributing functions (and thereby creating human tasks) can be clearly examined. In allocating tasks the designer needs to consider the feasibility of obtaining data from different sources and the desirability of doing so. Let us consider the next level of detail for Process 2. It takes in a dataflow of 'ValidUser' which is the output from process 1. It also takes in a data item CurrencyAmount from somewhere else and produces an output data item called 'Currency" which we may take to be the amount of foreign currency requested. The designer must now consider what the content of the data 'CurrencyAmount' will be. Some possibilities are;

- any amount expressed in the *local* currency (then the user would need to know the equivalent amount in the currency required)
- any amount expressed in the currency *required* (then the user would need to know what the equivalent was in local currency).
- a *restricted* amount in local or required currency (in which case, the system could offer certain amounts in the foreign currency and restrict the user to selecting one of these).

The designer needs to produce a more detailed level of description for process 2 in order to examine the alternatives (see Figure 2).

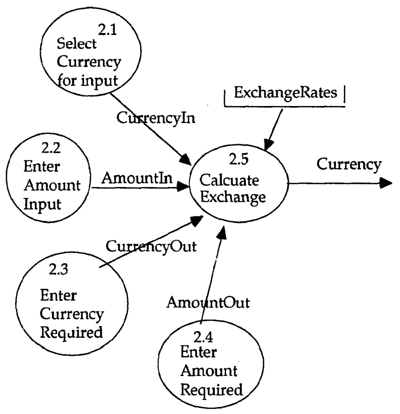

Figure 2 Level 2 DFD for process 2, Enter Requirements (first attempt)

Unlike Figure 1, Figure 2 includes a store of data which is (conceptually) necessary for the process to be possible. As we move down the levels of abstraction, we add details which are necessary at that level. Clearly process 2 will need access to some representation (a datastore[2]) of exchange rates; where that is located or how it is represented is not relevant to this level of discussion. Four possible processes are shown which are required for different scenarios. In one scenario the user's goal is 'I want the equivalent of £100 in DM'. Another scenario is 'I want 2000 Francs'. In the first case, the user performs processes 2.1, 2.2 and 2.3 of Figure 2 and in the second case, the user performs 2.3 and 2.4.

In allocating tasks, the designer will consider the cognitive load imposed on the user by any of the options chosen, how much learning will be required, what knowledge may be transferred from other tasks and how best to exploit that knowledge. The simplicity of task-action mappings are also important as is a simple conceptual-physical mapping. Task analysis techniques, particularly cognitive task analysis which focus on the user's task knowledge are appropriate at this point to inform the designer. Clearly in the allocation of tasks, one important consideration is the mental load demanded of the user by a particular user-system design. Indeed, we implicitly considered this above when arguing that the system should calculate the exchange amount. This is an easy task for the computer, but a difficult one for the human which is a good reason why the computer should perform it.

PHYSICAL DESIGN

Let us consider what a 'second attempt' analysis of process 2 in Eurochange might be like and impose the human-computer boundary as illustrated in Figure 3. Flows across the boundary indicate the user inputs and system outputs which are required. The system will display the currencies which are available and the user will select the required currency. The system will display the exchange rate between the requested currency and the user's local currency. The user will then enter the amount of the foreign currency required. The system will display the equivalent amount in local currency and the user may then accept this or return to the start of the transaction. Amounts are only allowable in certain denominations (depending on the required currency). If the user enters an incorrect amount the system will request that a more appropriate amount is entered. Clearly this solution is only one option and serves to illustrate the approach rather than defining the best design.

With the design illustrated in Figure 3 the designer has created three tasks for the human - 2.3 Select Currency, 2.4 Enter Amount Required and 2.8 Confirm Amount. It would seem clear that the user has (logically) to perform 2.3 (although even for this process automated options are available -see below), but there are many physical design options all of which accomplish this same, logical process.

If it is decided that the computer will support this process, then the currencies available can be pointed at by a mouse, selected from a list using cursor keys, displayed on a touch sensitive screen or displayed using 'hard' selection keys. The system could anticipate what the user requires either through a simple rule such as 'if the user is in France s/he will probably require French currency' or through more elaborate mechanisms such as inferring what the user requires from details of previous transactions which could be stored on the user's card (thus making it a 'smart card').

There are many other aspects to be considered. For example, in deciding the amount required, the user may need access to the exchange rate. We must consider the possibility of the user making errors and being able to correct them and of users changing their mind when they see how much they are asking for in the local currency.

CONCLUSION

This paper has provided a framework for HCI design which draws the important distinction between conceptual and physical design and places task analysis within that framework. If we adopt this framework then we must look for modelling mechanisms which are suitable for describing the whole human-computer system and which focus attention on the demands on users and machines which result from particular designs. Dataflow diagrams fulfil that function for the processing. Other techniques such as ERMIA are available for describing the conceptual structure of human-computer systems (Green and Benyon, 1995).

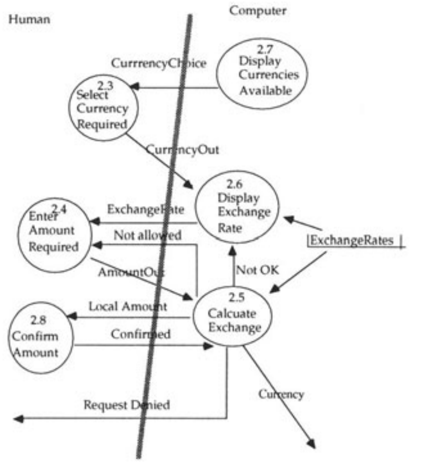

Figure 3 Second attempt DFD of process 2. Allocating tasks to Human and machine in the Eurochange example

The task allocation stage of developing human-computer systems is certainly one of the most important and one which will itself involve many iterations, prototyping of options, detailed analysis and user testing. The designer, starting off with a conceptual description of the whole human-computer system in the form of data models such as dataflow diagrams, considers each process bringing to bear task analysis techniques as appropriate. Sutcliffe (1991) makes a similar point, though with a different motivation.

It is certainly good practice to examine current human tasks as an understanding of these will inform the design process. However, it is equally important to abstract from current practice - in this case by focusing on the information flow required in the system. The method of using dataflow diagrams is effective and robust. Moreover, it conforms with software engineering techniques which will be used to implement the system and thus assists with the transfer of HCI considerations to mainstream software engineering.

ACKNOWLEDGEMENTS

The ideas presented here are developed from Part IV of (Preece, *et al.* 1994).

REFERENCES

Benyon, D.R. (1990) *Information and Data Modelling*, Blackwell Scientific Publications, Oxford

Benyon, D.R. (1992a) The Role of Task Analysis in Systems design in *Interacting with Computers* 4(1), 1992

Benyon, D.R. (1992b) The Discipline of Data in *Interacting with Computers* 4(2), 1992

Bösser, T. (1987) Learning in Man-Computer Interaction, Springer-Verlag

Braudes, R.E. (1991) Conceptual modelling: A look at system-level user interface issues. In *Taking Software Design Seriously* (Karat, J., ed.) Academic Press, London

Browne, D. (1994) *STUDIO Structured User-interface Design for Interaction Optimisation*, Prentice-Hall, London

Card S.K., Moran T.P., and Newell A.,(1983) The Psychology of Human Computer Interaction. Lawrence Erlbaum, 1983.

Carroll, J.M. Infinite detail and emulation in an ontologically minimised HCI in J.C. Chew and J. Whiteside (eds.) *Empowering People; CHI '90 Proceedings* 1990 ACM press

Checkland, P. B. (1981) *Systems Theory, Systems Practice.* Wiley

DeMarco, T. (1979) *Structured Analysis, Systems Specification.* Prentice Hall

Diaper, D. (1989) *Task Analysis for Human-Computer Interaction* Ellis Horwood, Chichester

Diaper, D. and Addison, M. (1991) Task Analysis and Systems Analysis for Software Development. In: *Interacting with Computers* vol 3(3) pp 124 - 139

Draper, S. (1993) The Notion of Task. In *Bridges Between the Worlds, INTERCHI '93* Conference Proceedings, adjunct proceedings (Ashlund S., Mullet K., Henderson A., Hollnagel, E. and White, T., eds), pp 207-8 Addison-Wesley, Reading Ma.

Fischer, G. (1989) Human-Computer Interaction Software: Lessons Learned, Challenges Ahead *IEEE Software,* (January) pp 44-52

Gould, J. D. and Lewis, C. (1985) Designing for Usability: Key principles and what designers think. *Communications of the ACM,* 28, 300 -11

Green, T. R. G. and Benyon, D. R. (1995) Displays as data structures: Entity-Relationship Models of Information Artefacts. In *Proceedings of Interact '95* North Holland, Amsterdam

Greenbaum, J. and Kyng M., eds., (1991) *Design at Work: Cooperative Design of Computer Systems.* Lawrence Erlbaum, Hillsdale NJ

Hix, D. and Hartson, H. R. (1993) *Developing User Interfaces: Ensuring Usability through Product and Process* Wiley, New York

Johnson, P. (1992) *Human-Computer Interaction: Psychology, Task Analysis and Software Engineering.* McGraw-Hill, London

Kieras, D. and Polson, P.G., (1985) An approach to the formal analysis of user complexity *International Journal of Man Machine Studies,* Vol. 22, 1985

Long, J, Cognitive Ergonomics and Human-Computer Interaction. In P. Warr (ed.) *Psychology at Work* Penguin, Harmondsworth. 1989

Moran, T. (1981) The Command Language Grammar *Intentional Journal of man-Machine Studies* 15(1) 3 - 50

Payne, S.J and Green, T.R.G, (1989) Task-Action Grammar: the model and its developments in Diaper, D. (ed) *Task Analysis for Human-Computer Interaction.* Ellis-Horwood

Preece, J. J., Rogers, Y., Sharp, H., Benyon, D. R., Holland, S. and Carey, T. (1994) *Human Computer Interaction: Understanding for Design.* Wokingham, UK; Addison-Wesley.

Rasmussen, J. (1986) *On Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering* Elsevier, Amsterdam

Shepherd, A. (1989) Analysis and Training in information technology tasks in Diaper, D. (ed) *Task Analysis for Human-Computer Interaction.* Ellis-Horwood

Sutcliffe, A.G. (1991) Integrating methods of human-computer interface design with structured systems development. *International Journal of Man-Machine Studies* 34 631-55

---

[1] A dataflow is any arbitrarily complex combination of data elements (which may include gestures, speech, text, mouse clicks, etc.) which travel together from one process to another.

[2] A datastore is any arbitrarily complex collection of data (which may be resident in the human) which is more persistent than a dataflow and does not travel between processes, but which is employed or updated by a process.