*Interacting with Computers vol 4 no 2 (1992) 246–259*

# Task analysis and system design: the discipline of data

**David Benyon**

Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

In the previous issue of *Interacting with Computers*, I offered a critique of task analysis (TA) arguing that, in general, task analysis techniques had failed to appreciate the significance of the shift in emphasis – towards logical modelling of the system and away from physical modelling – which had taken place during the early 1980s in information systems. As a result TA (if used on its own) would produce poor system designs because it fails to achieve sufficient device-independence (Benyon, 1992). In the commentary on this paper, Diaper and Addison (1992) defended TA in general and TAKD (e.g. Diaper, 1989b) in particular against the criticisms and made a number of assertions about my stance regarding humans and computers.

In this reply, I intend to clarify first what I mean by a data-centred view, second what I mean by device independence and finally why the data-centred view is a vital part of systems development. In doing this I hope to deal with many of the specific points raised by Diaper and Addison.

Underlying this discussion is a view of how system designers should undertake their designs and a theoretical, or philosophical basis of human–computer interaction (HCI) provided by the data-centred view. Whilst both of these are important subjects, there is neither the space nor the time to give adequate coverage here. However, I am sure that both the theory and the practice of HCI will benefit from this continuing debate on the preferred role of task analysis in system design.

My original concern about TA stemmed from the claim made by its advocates that 'task analysis is potentially the most powerful method in HCI . . . for producing requirements specifications' (Diaper, 1989a; preface). I believe this view is unsatisfactory because TA does not focus sufficiently on the *data* in the system. This paper is aimed at explicating this position.

The central part of this paper is an exposition of data models as conceptual models which capture both a human and a computer view of systems. This discussion is necessary in order to understand the notion of device independ-

Department of Computer Science, Open University, Milton Keynes, MK7 6AA, UK

 0953-5438/92/020246-14 © 1992 Butterworth-Heinemann Ltd

ence which is presented in the following section. Before turning to the details of data, it is important to obtain a perspective on system development.

## System development

There are many characterizations of how systems are developed and how systems should be developed. The model used in Benyon (1992) was chosen to emphasize the importance of conceptual modelling and task allocation rather than to prescribe a system development method. Many methods advocate a task allocation phase (such as Avison and Wood-Harper, 1990; Sutcliffe and McDermott, 1991; Damodaran *et al.* 1988). However, task allocation is not something that happens after conceptual modelling is complete. Task allocation is an extended and considered process of analysis, design and implementation. The allocation of tasks to human, to machine or to some combination prompts a discussion of both human and machine capabilities and resources. Performing task analyses here is an effective way of informing this process. No doubt errors in the analysis and omissions in the design will be exposed and the development process will cycle round to consider alternatives.

Models of system development are produced for different purposes, are more or less useful for those purposes and are more or less applicable to different systems. However it is clear that designers have to undertake some analysis work, some design – both logical and physical – and the system has to be implemented. There exists an uneasy relationship between these activities both in respect of the order in which they are undertaken and their respective contents.

Diaper and Addison (1992) view the stages of systems development as having identifiable inputs and outputs, but this is only one perspective. Analysis suggests design solutions. Design exposes holes in the analysis. Implementation, at least in the form of prototypes, is an aid to both analysis and design.

Data capture (or requirements specification), for example, is certainly something that has to be done thoroughly and thoughtfully and which is not easy. But what do we want to achieve? Simply *describing* the current system is useful only insofar as it reassures the users that the analyst understands what they, the users, do. It is *analysing* the current system that is important. Methods for data capture (such as task analysis) must demonstrate their analytic power. Obtaining appropriate descriptions of what people want to do and how they want to do it, coupled with how they should do it (e.g. for safety, security and integrity reasons) is vital for both analysis and design. Part of systems analysis is understanding the tasks which the system has to support. Structured methods of analysis and design are poor in both respects. It is for this reason that the systems view and rich pictures have been advocated in information systems design (Avison and Wood-Harper, 1990), TA has become popular in HCI and the urgent need to integrate HCI with information systems design has been recognized (e.g. Sutcliffe, 1989; Damodaran *et al.*, 1988).

Many techniques are available for both analysis and design and should be used as appropriate. Rather than trying to find an all-encompassing methodology, or the 'right' representation, the 'tool-kit' approach to systems development



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

(Benyon and Skidmore, 1987) recognizes that different situations demand different techniques. In this approach designers are trained in the use of different tools and techniques which they select as appropriate given their knowledge of the tool's strengths and weaknesses, personal preferences and the demands of the problem at hand. Task analysis techniques are a welcome addition to the analyst/designer's tool-kit, but they do not capture all the required views. An essential component of the tool-kit, or of any methodology, is one or more techniques for logical or conceptual modelling.

## Conceptual modelling

The purpose of conceptual modelling is to allow analysts to reason about a problem and come up with design solutions in the abstract. A good conceptual model is one that is appropriate to the situation at hand. It highlights significant information and suppresses unnecessary detail. 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. The constructs which a model employs, the ability to manipulate those constructs and the constraints which can be expressed are essential to its efficacy. Mathematics is a conceptualization which has been well used in many areas. Models made of plastic or wood are appropriate for conceptualizing physical structure. A good introduction to the principles of conceptual models in general and data models in particular is provided by (Tsichritzis and Lochovsky, 1982).

As HCI focuses on the interaction between humans and computers, a conceptual model of the human–computer system which is both an abstraction of humans and of the computer system would be desirable. On the human side, we may wish to model the conceptual structures which people have, the tasks which they perform, the likelihood of their making mistakes, the cognitive processing which they undertake, the learnability of the system, the knowledge required to use the system, and so on. On the computer side we may wish to model the structure of the computer system, the processing, the interface, the performance, the quality and so on.

We certainly do not want models to be tied to particular computer languages or programming paradigms (e.g. logic programming, object-oriented programming, etc) nor to particular individual people, because such models would fail in the objective of achieving an appropriate level of abstraction. The analytic, explanatory and communicative power of a conceptual model arises from the stucture, operations and constraints which that model is able to capture.

## Data models

I believe that it is unhelpful to see data models as models *of* data. Data models are models of some aspect of the system, in our case a human–computer system, which are *made* of data. The object of the model is the human–computer system;



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

data is the material from which the model is constructed. The contention here is that using data as the building block of models is particularly appropriate for human–computer systems.

The basic structural component of a data model is a data item. A data item is a semantic object consisting of one or more symbols, a name (and usually a more comprehensive description of the meaning of the data item) and a context. The name, description and context ascribe the semantics to the data item. Instances of data items – the actual and potential values – are conveniently generalized by the data item's name. Details of data items and their meaning can and should be stored in a data dictionary, thus making the semantics explicit. Resolving conflicting understandings and meanings for the same data item, or for the same name of a data item is a major part of the analyst's work.

For example an analysis of an electronic mail system, may reveal that there is a data item type which the analyst, in consultation with the users, decides to call *WhoTo*. This data item may be defined as the input which is entered in response to the mail system prompt.

**To:**

indicating the recipient of a message. The instances of this data item include

**rpt@uk.ac.leeds.dcs**
**@Researchgroup**
**T_Bolton**
**Earn-relay"John@ibm.com"**

Data items are established by the analyst through data capture techniques such as interviewing, observation, recording, prototyping, searching and so on. Analysts do not simply perceive the world, they interact with the existing system, or descriptions of it, and decide to represent certain aspects using certain data items. Analysts must take great care to identify implicit as well as explicit data. The height of a pile of boxes in a warehouse or the whirring of a disc drive are just as much data as a user's name or a product code.

Using this basic component, data models impose a discipline. In the first instance, the meaning of each data item has to be established. In the example above, the meaning of *WhoTo* is far from clear. Analysts must consider the structure of the data item, the allowable values of the data item and whether there is more than one type of data item. In this case, the analysis would reveal that

- **rpt@uk.ac.leeds.dcs** is an address on the JANET network, the syntax is *UserId@Address* and that *Address* is made up of *Country.TypeOfOrganisation.Institution.Node*. This type of *WhoTo* in fact consists of five distinct data items.
- **@Researchgroup** is a distribution list which actually refers to a file called *Researchgroup.dis* consisting of



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

any number of individual data items of the type *WhoTo*.

- **T_Bolton** is a user on the same node of the network as the sender.
- **Earn-relay"John@ibm.com"** Includes a *NetworkName* data item (in this instance, 'Earn-relay') and that is followed by an address of the form *UserId@Node.Institution.TypeOfOrganization.Country* (though *Node* and *Country* do not appear in this instance).

This analysis has already highlighted the existence of a number of other data items – *NodeName, NetworkName, Country* and so on – each of which will require further analysis. This analysis will subsequently reveal the wealth of detail about names and addresses that permeate electronic mail systems.

An important modelling technique is that of generalization and specialization. We recognize that all the data items listed above are of a certain *type* as demonstrated by the fact that they are all input in response to the 'To'; prompt. The type of data item is a generalization of the specific instances. However, we have also recognized that there are further types within this – the people on the same network type, the people who have a network name as part of their *WhoTo* etc.

The next level of data analysis considers the relationships which exist between data items. All data items have relationships to other data items. Analysts have to establish which relationships are important for the system under consideration. The relationships which are of particular interest are the mappings between the actual and potential values of the data items (Benyon, 1990). The technique of normalization enables the analyst to explore the semantics of the data by examining these syntactic relationships. On the basis of this well-defined and rigorous technique, data items can be grouped together in a way which clearly expresses the semantics which hold in a particular context.

In the e-mail example, analysts must consider the relationship between the data items *NodeName* and *NetworkName*, for example. Clearly a value of *NetworkName* may be associated with many values of *NodeName*, but can a value of *NodeName* be associated with one or more values of *NetworkName*? The semantics of this relationship concerns whether a node can be on more than one network or not. This depends on how *NetworkName* and *NodeName* are defined. The analyst is thus forced to go back to the users and discover how these data items will be defined. This in turn will determine the relationships which they have with other data items.

Another level of abstraction within the data modelling paradigm is provided by an object which is usually known as an 'entity'. An entity is an aggregation of data items, expressing the semantics that certain data items belong together. Entities may be derived from the normalization process, or they may be determined in a top-down manner by analysts using their experience. These entities can then be examined in terms of the relationships which exist between



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

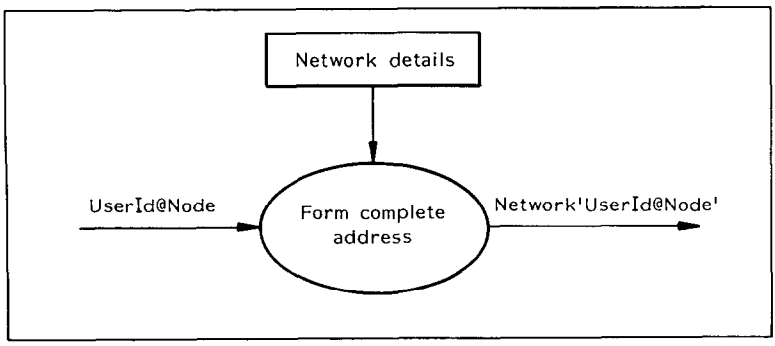


*Figure 1. Example dataflow diagram*

them. The most usual technique here is the entity-relationship (ER) model which is an excellent analytic tool as well as a design specification.

In the e-mail example, we may discover entities such as User, Node, Network, DistributionList, TypeOfOrganization and so on. Part of the discipline imposed by ER models is that every entity must have an identifying data item. The choice of identifier dictates what the entity means. For the entity, User, for example, if I choose *UserId* (such as T_Bolton, John or rpt) as my identifier it means something very different than if I choose *UserId@Node* (T_Bolton@uk.ac.open.acs.vax, John@IBM.COM or rpt@uk.ac.leeds.dcs) or *Network"UserId@Node"* (e.g. Earn-Relay"John@IBM.COM"). In choosing an identifier (and hence defining the meaning of the entity) analysts must ensure that there is a close correspondence between the entity and the concept(s) employed by users.

Another way to model the system is to look at how data flows between processes. A process, or functional, model made of data concentrates on the data that is strictly necessary for processing to occur. This may be contrasted with a model of the physical system (such as a flowchart) which models the movement of physical objects such as documents. The discipline imposed by a data-centred process model (such as the dataflow diagram, DFD) is concerned with the completeness and consistency of the description. By casting the model in terms of data, processes can be examined to ensure that they have access to enough data to complete the transformation of the data entering the process to the data leaving the process.

The DFD shown in Figure 1 illustrates this property of data-centred process models. If the input to the process 'form complete address' is just the data item(s) *UserId@Node* and if the full address required by the system is *Network"UserId@Node"* then the network details have to be provided from some store of data. Coupled with the requirements for precise definitions of the data items involved, the DFD can be checked for completeness and for logical processing capability. For example, is it possible to access, automatically a list of networking details given the data *UserId@Node* or is this necessarily a human activity?

Data models impose a discipline on the processes of analysis and design. They demand precise definitions of data items to be formulated. They provide



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

rigorous techniques for aggregating data and establishing semantically sensible data objects. They provide analytic techniques and force the designer to ensure that the processing of data is logically possible.

## Data models, humans and computers

One of the criticisms levelled at the data-centred approach by Diaper and Addison (1991) is that it is 'computer-centric' by which they mean 'more suited to modelling computer systems than people' (p. 128). In making this claim, they fail to distinguish between *infological* data models which are concerned with establishing a user-orientated model of the system and *datalogical* models which are concerned with representing the computer view (Sundgren, 1979; Benyon, 1990). Conceptual data models may capture both the infological and the datalogical perspective.

In the previous section, I emphasized that data is a semantic concept and as such it is an appropriate mechanism for modelling humans. The underlying assumption of cognitive psychology is that the human is an information processor. But information only becomes available when data items are structured and associated with each other. Indeed it is the way that data items are associated which enables information to be derived.

For example, a piece of data (i.e. a value of a data item) such as '13.50' is meaningless until it is associated with a name or description such as 'PriceIn-Pounds' or 'DepartureTime'. We are now able to obtain some meaning from the data, but it still provides no information until it is associated with another data item such as 'TrainJourney' and a value for this item such as 'Leicester to Nottingham'. The relationship type such as

a TrainJourney has a DepartureTime and a PriceInPounds

should approximate to the human concept (of a train journey, in this case). This is the infological model.

Data can also represent the computer data structure (the datalogical model). An instance of the relationship is provided by appropriate values of the data items, for example,

| *TrainJourney* | *DepartureTime* | *PriceInPounds* |
|---|---|---|
| Leicester to Nottingham | 13.50 | 14.20 |

Notice that without the data descriptions, it would be unclear which value was the price and which was the departure time. Context is also important. In this case we need to know when this relationship is valid in order to obtain information.

Exactly what information is derived depends on the previous knowledge of the user, the semantics which they attribute to the data items and the relationships which are perceived. This is one reason why data modelling is so vital; the semantics of, and relationships between data items are made explicit.



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

If the conceptual data model is carefully designed it will have a close resemblance to human conceptual structures. The ER model, being an infological data model, represents a user-oriented view of the system. Entities are created by the analyst in order to model the world. 'Concepts are inventions of the human mind used to construct a model of the world' (Sowa, 1984; p.344).

Entities should map onto the concepts employed by people. They also map onto computer data structures. Thus starting from a shared conceptual data model, the technical system designer can develop the computer implementation and the interface designer can specify a system structure that reflects users' concepts.

Information is derived from data, hence humans, at one level of abstraction, are data processors rather than information processors. Computers are also data processors. All computer systems can be considered as (datalogical) data models. They use data to present some aspect of the world as interpreted by analysts in consultation with users. The big difference is that humans (or more generally, agents) are task-centred as well, whereas computers are not. Computers are process-centred.

By task-centred I mean that agents undertake activities which they believe are necessary and/or desirable for their purpose. Computers, in general, lack the sense of purpose. In designing computer systems we impose a task structure onto the computer system to make it easier to use. We organize the processes into some structure which reflects the tasks which humans want to perform. The problem with imposing a task structure is that a complex relationship exists between data, users and tasks. The same data is used in many tasks, different users have different tasks, different users use different data for different tasks and new tasks emerge which could not have been anticipated during the analysis and design of the system.

For example, within the e-mail system, I store the messages which I have received in various 'folders' (directories). This data can be used to accomplish a task such as 'display the messages which I have received since November 1991'. It is not easy but it can be used. On another occasion I may have a different task such as 'I need to find the e-mail address of that person who mailed me from France about the Marseille conference'. Since I have saved my messages, I know that the data exists, but because it is structured according to task and this task has not been anticipated, I cannot accomplish my task.

Data models represent the semantics of the system. They perform several vital functions. First they provide an abstraction of human cognitive structures. They represent the concepts which people have, or must understand, in order to use the system. Second they discipline designers by forcing the semantics to be made explicit. Third, they can be used to structure the data in the computer in a way which is independent of task, allowing flexibility for users to develop their own task structures. In this sense data models bridge the 'anthropocentric gulf'.

There is certainly a computer-centric view of data models, but it is not the only view. For example, Green (1991) uses ER models to represent human conceptual structures. Most texts on structured methods fail to recognize, explicitly the infological realm of data models and this leads them to offer poor definitions of the concept of an entity.



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

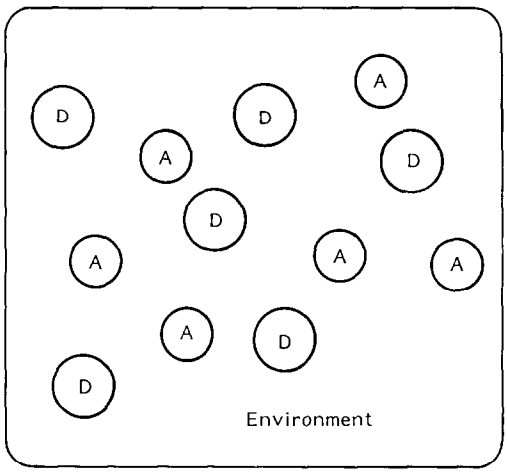


*Figure 2. The world can be seen as consisting of agents (A) and devices (D) constrained and influenced by their environment*

## Systems, tasks and device independence

For the purpose of focusing attention on human–computer interaction, I propose that we may view the world as consisting of two types of entity – agents and devices – which exist within the constraints and influences of an environment. This is illustrated in Figure 2. Both agents and devices are systems (relatively complex 'wholes'). Agents are intentional autonomous systems in that they have beliefs and desires and can formulate their own goals. People are agents. Agents undertake goal-directed activities by making use of devices.

We can define a 'goal' (or external task) as a state of the environment or of the agent which is preferred by that agent to the current state. For our purposes, it would be helpful to exclude 'impossible' goals so we can add that an agent must believe that the goal is achievable within the environment. An agent formulates goals based on its desires and knowledge of the devices and agents which it perceives as existing in the environment. Goals are thus constrained by the agent's *weltanschauung*. For example, I do not formulate the goal of visiting another galaxy because I know of no combination of devices and agents whch will enable me to do this.

Given that the agent has formed a goal, the agent then selects a device which will enable it to achieve that goal. The process of selecting a suitable device depends on how the agent conceptualizes the device. The agent does this at one or more of three levels (following Dennett, 1987). First, the agent may consider the intentional stance – whether the perceived purpose of the device is appropriate for the purpose (goal) at hand. For example, if my goal is to communicate with another person, then an oven is a device which would probably be rejected on the grounds that I perceive the purpose of an oven to be the cooking of food rather than to communicate with another person.

Second, the agent can consider the design stance. The telephone system,



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

e-mail, writing and sending a letter and shouting loudly are all devices which are appropriate for the purpose as they can all be used to communicate with another person. Each of these has certain logical capabilities (the ability to store a message and to transmit it through time or space), but each also has certain logical constraints which derive from the design of the device. Telephone systems, because of their design, require access to a telephone. Writing and sending a letter imposes time constraints. It is a logical consequence of the way that the writing-and-sending-a-letter system works that the letter will take time to reach the recipient.

Finally the agent may conceptualize the actual interaction with a specific device (the physical stance). Logically, e-mail requires access to a computer. Physically, it means using a particular computer (at a particular time and in a particular place) and the device may be perceived as suitable or not when considered from this physical perspective.

Once the device has been selected, weighing up these different considerations, the agent begins to interact with the device. Up until this point, there has been no (actual) interaction. The process of selecting a device has been based on the agent's conceptualization of the device; the agent's mental model of some future interaction.

Given this view of interaction, we can now define a 'task' (or internal task) as a goal plus a device. The goal can be accomplished using a variety of devices. Once a device is selected, the tasks necessary to accomplish the goal are prescribed by the logical structure and functioning of the device, i.e. by the way that it has been designed, or has evolved.

This analysis may be applied at any number of levels of abstraction. Once an agent has started to interact with a device an agent–device (human–computer) system is established. This is constrained and influenced by the environment which consists of other agents (e.g. advisory staff) and devices (e.g. documentation). Within this system, the agent will formulate goals and select devices to accomplish those goals. For example, the agent, having selected e-mail as the device to communicate with another agent, now formulates the goal of entering the text of the message. The devices available for this include editors, word processors or typing the message in directly. These are evaluated according to the agent's conceptualization of their suitability for the purpose, their logical structure and functioning and their physical characteristics. A particular device is selected, thus establishing another agent–device interaction.

At some point, the agent physically interacts with a device by performing an 'action'. That is to say, the agent transmits some signals which the device is capable of receiving (e.g. the agent types a command on a keyboard). The agent then has to evaluate any signals transmitted by the device (or some linked device) and decide to what extent the agent's goals have been met. Conceptual interaction is based on the agent's models of the devices. Physical interaction concerns phenomena observed by the agent and signals transmitted by the agent.

This analysis highlights three levels of description of agent–device interaction. Level 1 concerns purpose. An agent evaluates devices on the basis of how appropriate they appear to be for the agent's goal. At the conceptual or logical



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

level –level 2– the design of the device contrains what the agent has to do and describes what the logical capabilities of the device are. The third level is concerned with how the device does something physically.

These three levels reveals two levels of device independence:

- *logical device independence*, between levels 1 and 2, is a separation of the purpose of the agent-device interaction from the logical description of how it is to be accomplished (as determined by the type of device).
- *physical device independence*, between levels 2 and 3, is a separation of the logical description from the requirements of a physical interaction (as determined by the actual device).

It is level 2, the logical or conceptual level which is vital as it sits between levels 1 and 3. The logical level has, first, to be an appropriate structure and have appropriate functions in order to fulfil its purpose, and second, it dictates certain characteristics of the physical design. The physical characteristics of the device must support the logical structure and functions.

For example, an oven is a poor device for communicating between agents because it lacks certain inherent functions such as the ability to transmit messages. If someone were to tell me that an oven was a suitable device for communicating with another person, then I could be expected to question exactly how the physical construction of the oven could support the logical requirements associated with my goal. Where are the objects which can represent my message? Where is the functional capability to transmit?

This is not to say that a device cannot be used for novel purposes. However, the design does constrain the purposes to which a device can be put. Similarly there can be novel implementations of the design, but any implementation must support the designed structure and functions.

Goals, or external tasks are dependent on the environment and the agent's conceptual knowledge of the devices (and other agents) available. Internal tasks are dependent on the design of the device. The (type of) device dictates the tasks which the agent has to undertake. Actions are dependent on the actual, physical device (and other constraints such as time and place).

Conceptual data models are representations of the logical device. Task analysis is a model of HCI (or agent–device interaction) which focuses on the mappings between two or three of the three levels; goals map onto internal tasks which map onto actions. In focusing on this, TA is a vital part of system development. A straightforward goal-task-action mapping will certainly improve the human–computer interaction. But because TA focuses on the mappings, it is necessarily device dependent. Indeed that is surely the *raison d'etre* of TA.

## Conclusion

The purpose of this paper has been to explicate a particular view of human–device systems. Data is the most important and probably the only thing which humans have in common with computers. By building models made of data we



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

establish a representation which is common to both agents and devices and avoid making physical design decisions too early. Conceptual data models are particularly appropriate for representing the logical level of a device. In addition to the logical level we have recognized the existence of a physical level of description and a level concerned with (perceived) purpose.

If we accept this description of HCI, then we have to acknowledge that the development of agent–device systems involves four major activities:

- declaring the purpose(s) of the system
- defining a logical structure and functions which can accomplish the declared purpose(s)
- deciding which elements of the agent–device system will perform which functions and provide the necessary structural components
- physically developing devices which are capable of supporting the logical structure and functions and ensuring that agents have the appropriate knowledge and skills to perform their actions.

These activities are interdependent, iterative and may occur simultaneously. The results of these activities require constant prototyping, evaluation and refinement.

There is a wealth of tools and techniques to assist analysts and designers in their tasks. Techniques for establishing purpose include the CATWOE method of the soft systems methodology (Checkland, 1981) and the rich pictures of Multiview (Avison and Wood-Harper, 1991). Methods from systems analysis and psychology such as interviewing, protocol analysis, card sorting, questionnaires, personal construct psychology and document analysis can be used for many purposes. Prototyping, formal evaluation techniques, simulations and animation similarly are applicable at different times. The large numbers of TA techniques may be used at various points in the process. Data-centred techniques provide another perspective.

The question which needs to be addressed with regard to any technique is; how appropriate is it for the purpose at hand? This depends on the nature of the system, the nature of the analyst, the level of description being sought, the constructs and operators provided by the technique and its usability (ease of use, clarity, learnability, flexibility etc.) with respect to the purpose to which it is being put.

It is always difficult to separate the effect of a technique from the influence of the analyst. A good analyst will achieve a good analysis almost despite the technique used. It is the average analyst who gains most from a well-defined and rigorous technique. Diaper and Addision are, no doubt, good analysts and are able to arrive at novel solutions using TAKD. The technique has yet to prove that it can be used effectively by average analysts and that it can produce similar analyses of a situation by a variety of analysts. Data-centred techniques, on the other hand, do produce this type of consistency.

TAKD includes a number of tools. The TDH, since it is a hierarchy (whether a 'true hierarchy' (Diaper, 1989b) or used in its weak sense (Diaper and Addision, 1991) can only represent explicitly one side of a relationship or it must include



Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

unnecessary duplication in its descriptions. (A directed graph is just as bad. I may want to go in a different direction! As a contrast, the ER diagram is a non-directed graph.) A hierarchical representation may be suitable once a system has been designed (i.e. to represent the physical device level), but is restrictive during analysis.

TAKD incluces no definition of its objects or details of its actions. Johnson (1992) provides an example of TAKD for a computer-assisted jewellery design system which includes the generic objects: 'Setting', 'Orientation' and 'Jewel choice'. Although I have an intuitive understanding of what these things are, no precise definition is offered and, I suspect, confusions will arise. In contrast, entities demand rigorous and unambiguous definitions.

Diaper and Addison (1991) claim that TAKD is suitable for arriving at the conceptual level through the 'generification' technique, and this may be true. However, the need for the analyst to be 'insightful and creative' (Diaper, 1989b; p.136) and the admission that 'It is difficult to produce a heuristic, or even advice, for how to do this vital stage' (p.137) suggests that TAKD does not provide sufficient support for this activity. Data models provide well-defined and exacting techniques for the generification process.

TAKD certainly prescribes a thorough method for *describing* the current system. It is user-centred and focuses on the tasks which people perform and in this is complementary to other methods which focus on purpose, concepts, functions, security and so on. But I am yet to be convinced of its *analytic* power because the constructs, operators and constraints which it employs do not appear to provide that power. It is for this reason that I dispute the claim that TA is the most powerful method in HCI for requirements specification.

There are doubtless situations in which task analysis will be effective, but there are many situations in which it would appear to produce a limiting and overly device-dependent design. There are other techniques arising from work in information systems which are likely to be more effective which can offer a representation of both human and computer which is independent of task and device. HCI designers should be aware of these alternatives and the power which they offer.

## References


**Avison, D. and Wood-Harper, T.** (1990) *Multiview: An Exploration in Information Systems Development.* Blackwell Scientific Publishers

**Benyon, D.R.** (1992) 'The role of task analysis in system design' *Interacting with Computers* **4**, 1, 102–123.

**Benyon, D.R. and Skidmore, S.** (1987), 'Towards a tool-kit for the systems analyst', *Comput. J.* **28**, 1, 2–7

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

**Benyon, D.R.** (in preparation) 'A semiotic model of interacting systems' Submitted to *IEEE Trans. Systems, Man and Cybernetics*

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

**Damodaran, L., Ip, K. and Beck, M.** (1988) 'Integrating human factors principles into





Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018

structured design methods' in **Bullinger, H-J., Protonotorios, E.N. Bauwhuis, D. and Reim, F. (eds)** *Information Technology for Organizational Systems* North Holland

**Dennett, D.C.** (1987) *The Intentional Stance* Bradford Books, MIT Press

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

**Diaper D.** (1989b) 'Task analysis for knowledge-based descriptions (TAKD): the method and an example' in **Diaper, D. (ed.)** *Task Analysis for Human–Computer Interaction* Ellis-Horwood

**Diaper, D. and Addison, M.** (1992) 'Task analysis and systems analysis for software development' *Interacting with Computers* **4**, 1, 124–139

**Green, T.R.G.** (1991) 'Describing information artifacts with cognitive dimensions and structure maps' in **Diaper, D. and Hammond N. (eds.)** *People and Computers VI* Cambridge University Press

**Johnson, P.** (1992) *Human–Computer Interaction* McGraw-Hill

**Sowa, J.F.** (1984) *Conceptual Structures* Addison-Wesley

**Sundgren, B.** (1979) 'Database design in theory and practice' in **Weber, H. and Wasserman, A. (eds)** *Issues in Database Management* North-Holland

**Sutcliffe, A.G. and McDermott, M.** (1991) 'Integrating methods of human–computer interface design with structured systems development' *Int. J. Man Mach. Studies* **34**, 631–655

**Sutcliffe, A.** (1989) 'Task analysis systems analysis and design: symbiosis or synthesis? *Interacting with Computers* **1**, 1, 6–12

**Tsichritzis, D.C. and Lochovsky, F.H.** (1982) *Data Models* Prentice Hall

Downloaded from https://academic.oup.com/iwc/article-abstract/4/2/246/697945 by guest on 23 November 2018