Integrating Human Computer Interaction with Jackson System Development

A. G. SUTCLIFFE* AND I. WANG†

* *Department of Business Computing, City University, Northampton Square, London EC1V 0HB*

† *BIS Banking Systems*

Procedures for integrating task analysis and design of the human computer interface with a structured system development method, Jackson System Development (JSD), are described. Process Structure Diagrams, the major modelling notation of JSD, were used to specify tasks which were analysed to predict cognitive complexity. Task complexity analysis helped allocation and design of human tasks to produce task support specifications. The cognitive dimension of task analysis highlighted the need for task support actions, especially design of information displays to support working memory.

Dialogue specification for dialogues and interface displays were based on the task analysis and information requirements. Further analyses addressed the timing constraints on human and computer actions, and derivation of display requirements from task-related information needs. Preliminary evaluation of the method showed the method proved easy to learn even for non HCI specialists.

Received October 1990

1. INTRODUCTION

Human Computer Interaction (HCI) is acknowledged to be an essential component software development[19] yet the practice of HCI in software engineering is minimal.[3,4] One reason for the poor spread of the HCI principles and practice is lack of integration between HCI research and practice with methods in software engineering.

Many HCI specification and design methods have been proposed[12,15,18] but these have paid little or no attention to system development methods (see Wilson *et al.*[35]). Likewise, authors of system development methods have not considered design of the human computer interface, as can be seen in the study of method concepts by Olle *et al.*[21] More recently integration of HCI and software engineering methods has been recognised as a necessary goal.[29] It is the contention of this paper that practice of good human computer interface design will only result from integration of HCI principles and procedures within existing system design methods. By using the notations and tools familiar to software engineers, the HCI community may influence the creators of human computer interface software by supplementing their methods with good HCI practice.

It is beyond the scope of this paper to propose a complete methodology of HCI design, consequently only two topics will be considered: task analysis and dialogue specification. Experiences in adapting a commonly practiced structured system development method, Jackson System Development,[10,26] for task analysis and dialogue design are reported. Brief details of JSD are presented before extensions to the method for HCI specification are described. Use of the method is illustrated with a banking application.

2. JACKSON SYSTEM DEVELOPMENT AND ITS MODELLING NOTATIONS

JSD is an entity life-history oriented method which makes it appealing for HCI specification since tasks and user-system dialogues can be modelled as event sequences which happen to objects. One objective of this study was to use JSD, or more specifically its notations, for the purposes of HCI specification whenever possible. JSD makes extensive use of structure diagrams for process specification. These diagrams describe event sequences in terms of three control primitives: sequence, selection and iteration; thereby expressing a process structure which becomes the template for a program design.

JSD attaches considerable importance to time ordering in specifications. JSD starts by describing entities which are modelled as a life history of events. Fig. 1 illustrates a life history of a foreign exchange transaction from its

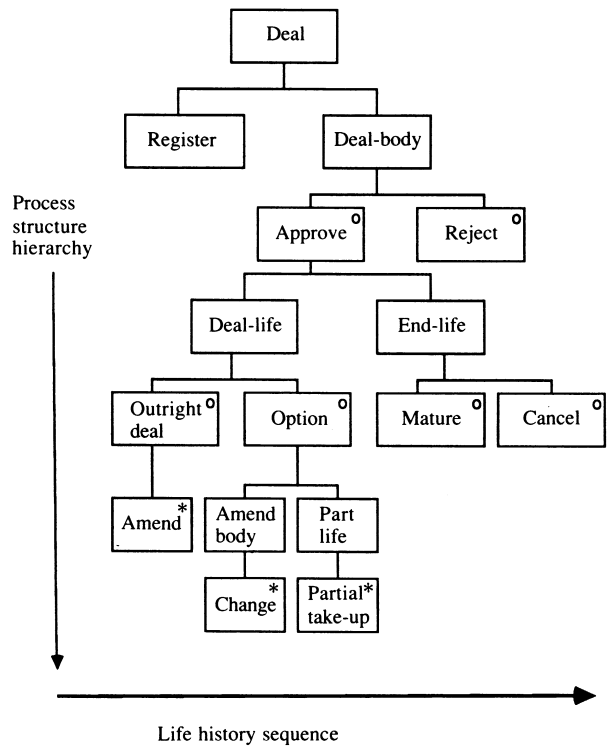

Figure 1. Process structure diagram: deal entity.

inception through to maturity. The process structure diagram shows a series of actions which happen to the Deal entity in a time order starting with Register and ending with either Mature, Reject or Cancel. The second main JSD notation is the System Specification Diagram (SSD). JSD describes systems as an asynchronous network of concurrent processes. All significant events to which the system must respond are modelled with datastreams. Another communication construct, the State Vector, is provided to model access to a process's data by another, inspecting process. JSD advises that the system specifications should be built by first modelling the entities and then adding function processes for system functionality such as input validation, algorithms and report/output generation.

2.1. Approach

The intention was to design user-system dialogues and information displays to support user's tasks and to take into account the limitations of human information processing. The prime motivations were to explicitly link analysis of users' tasks to design of the human computer interface; to introduce cognitive analysis of user-system interaction into the design process; and to produce a method for defining interface displays. Display design, in particular, has been ignored by both HCI and SE methods. Other concerns, which indicated the need for analytic techniques, were specification of timing constraints on interactions and design to accommodate human cognitive limitations, such as working memory.

The approach taken was to establish the specification components for user-system interaction which an integrated HCI-SE conceptual model would have to capture, The HCI literature was then reviewed by appropriate techniques and procedures advised by HCI methods. Then JSD and its notations were investigated for the purposes of HCI specification.

3. TASK ANALYSIS

The objectives of task analysis are open to many interpretations (see Bailey,[1] Wilson *et al.*[35] for a review). Two common themes are specification of functionality and analysis of cognitive complexity.[15] Tasks in HCI are a description of goal-related human activity. Task analysis describes human activity as procedures composed of actions and has similarities with functional decomposition espoused by structured systems analysis methods.[6] However, task analysis has the additional objectives of analysing human performance and the knowledge necessary to carry out an activity. Cognitive complexity evaluates tasks from the perspective of human psychology. By attempting to specify how mentally difficult a task will be, the difficulties in learning a task can be anticipated.

It was not the intention to create a sophisticated cognitive task analysis,[2] since the aim was to create a simple, easy to use method of immediate practical benefit for systems analysts.

JSD notation was used to describe tasks in terms of sequences of actions. Process structure diagrams (PSDs) provided a suitable means for task specification because actions are modelled in an event sequence which can map to task steps or components of user-system conversations. As PSDs were selected to represent tasks, the investigation continued to see what other aspects of HCI specification could be added to this notation.

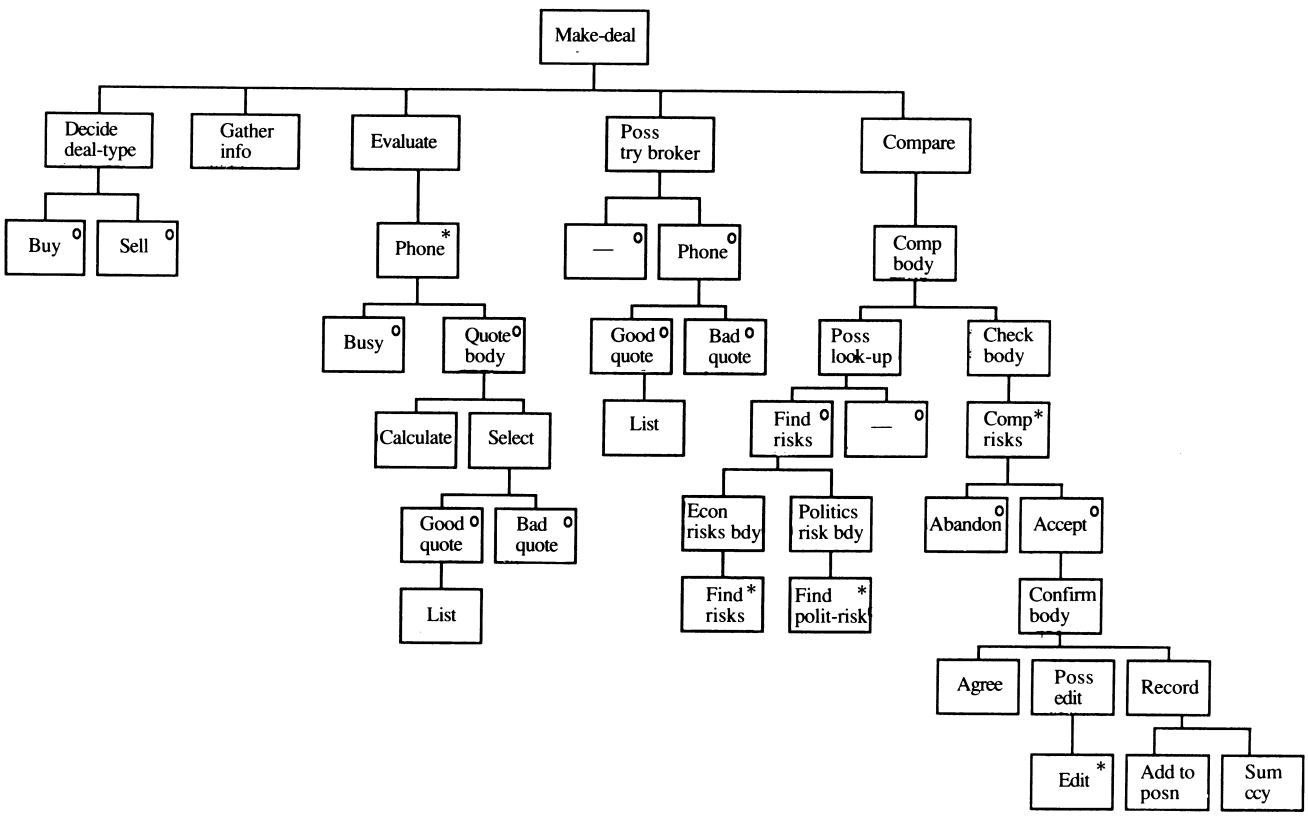

Figure 2. Initial task description: make deal.

3.1 Task analysis

The scope of investigation in systems analysis is inevitably influenced by the potential for automation. Human activity, such as negotiating a selling price for a deal over the phone, is usually considered to be outside the Software Engineering (SE) conceptual model. However, a complete task analysis should consider specification of human as well as computer activity and no pre-emptive decisions should be taken about allocation of activities to people or computers. Hence the first change brought about by the addition of HCI is to widen the scope of the system investigation.

The identity of some tasks is implicit within the event/actions in JSD entity model, for instance Register, Cancel and Mature a Deal (see Fig. 1). However, JSD event/actions in the entity model do not describe procedures, consequently the human activity necessary to carry out the task which result in a life-history event (e.g. Register on the Deal entity) is described in JSD function processes. This is consistent with JSD philosophy that entity and functional modelling should be separated.

The process structure diagram (PSD) notation easily accommodates task modelling. Task process structures, more properly termed 'interactive functions' in JSD, describe human activity as a time ordered sequence of actions. Judgement has to be exercised when modelling human activity because JSD states that actions are conceptually instantaneous, essentially an event. Some actions, e.g. Evaluate deal, were indeterminate and did not match the JSD concept. The definition of JSD actions had to be relaxed to allow inclusion of human oriented, less determinate actions. As granularity of the definitions of actions is a matter for the analyst's judgement in JSD, this change does not infringe the

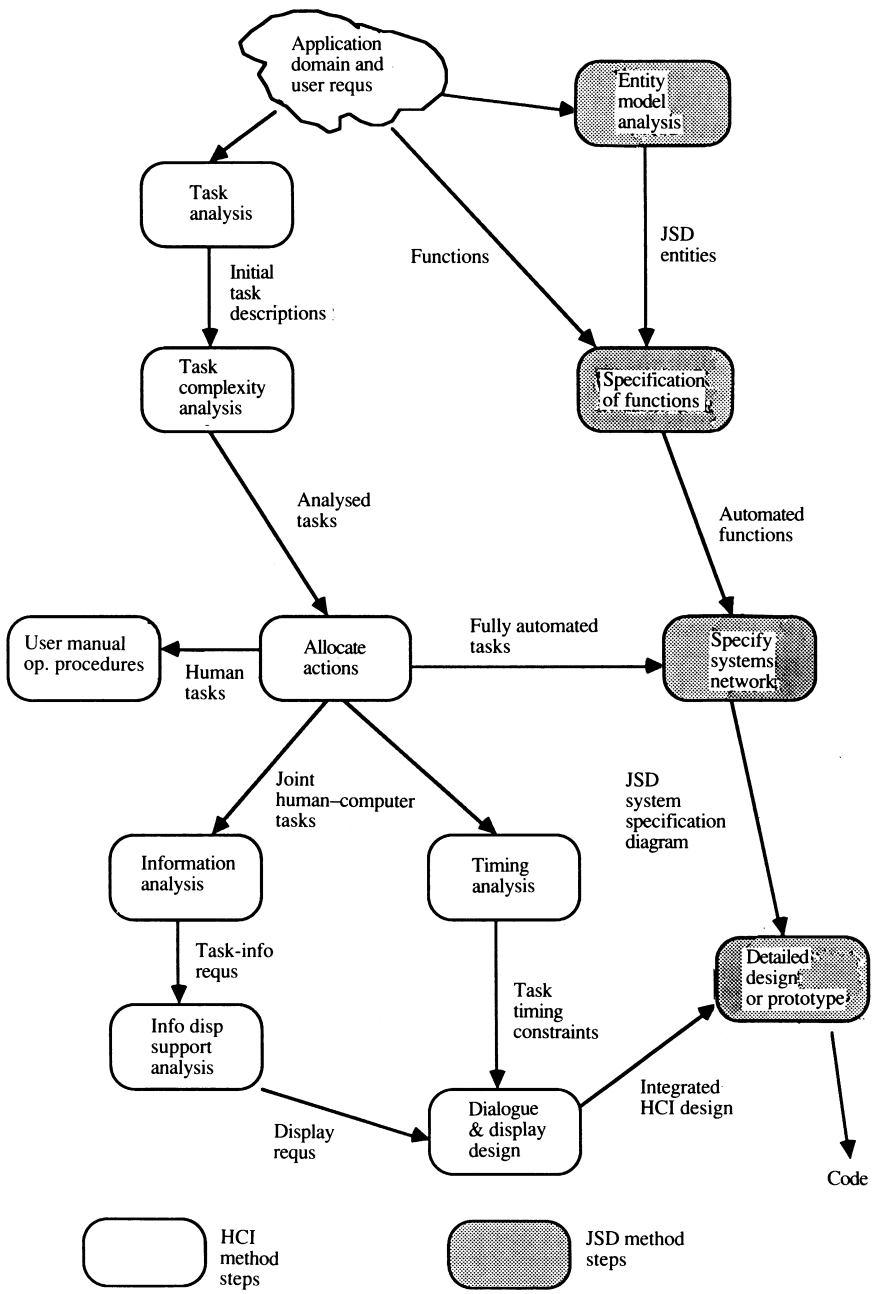

Figure 3. Summary of steps in JSD-HCI method.

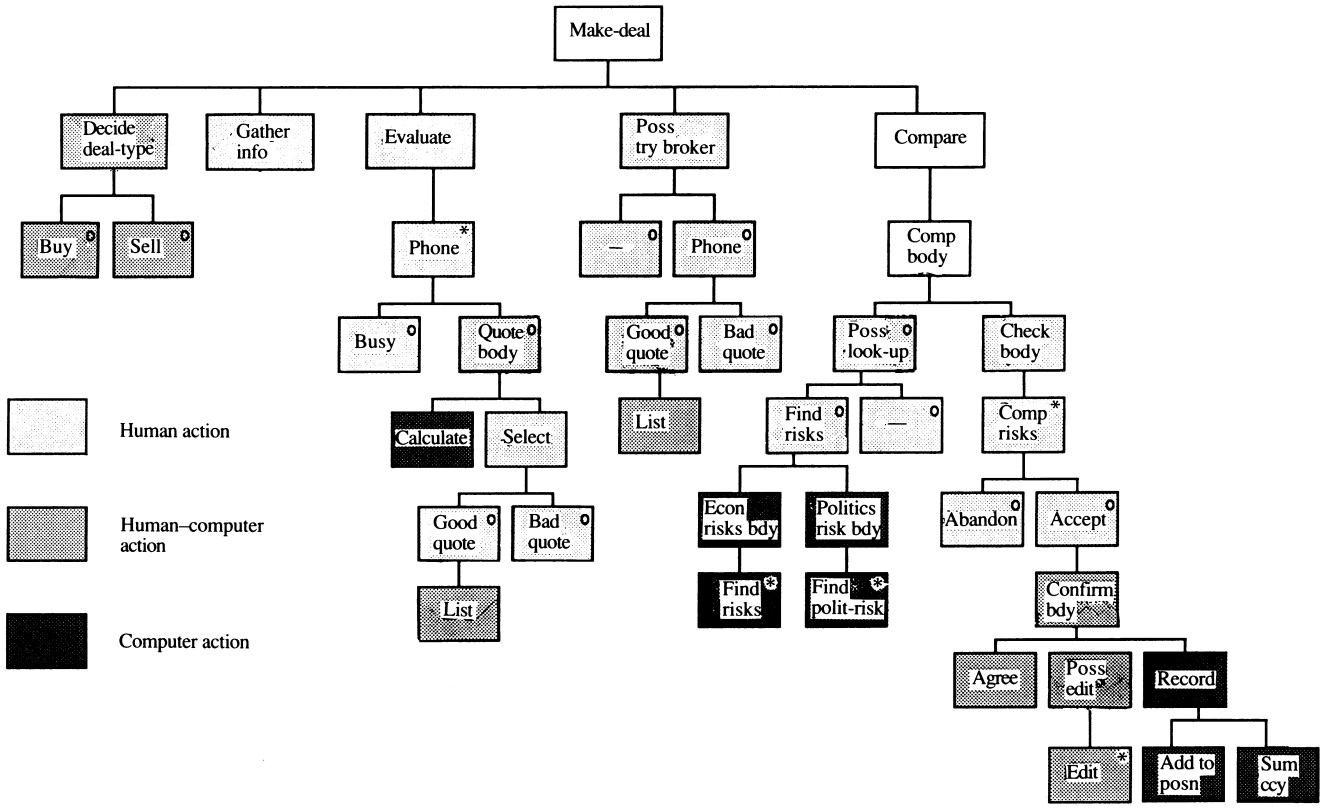

Figure 4. Allocation of actions to human or computer for make deal task.

method's principles. Actions performed by people are frequently not deterministic because different individuals do actions in different ways. These actions are unlikely to be computerised, so the impact of less precise specification on software quality is minimal. The initial task description of the *Make-Deal* task is shown in Fig. 2.

After initial description, tasks were allocated to either users or the computer system according to well established human factors criteria, e.g. algorithmic verses heuristic reasoning, need for judgement, quantity and quality of date – see Bailey,[1] Sutcliffe,[27] for further details. As many tasks involved both human and computer activity, allocation was performed at the action level. Each action was assigned as a system or user action, or a joint cooperating action which required further elaboration to specify computer support for human actions. The output from this step was two sets of PSD diagrams. One set described users' tasks which were not to be computerised and the other set describing JSD functions either fully computerised with no interaction (fully automated tasks) or interface functions for semi automated tasks. The steps in the extended method are illustrated in Fig. 3.

An example of an interface function is shown in Fig. 4. Note that the structure is based on the initial task description (Fig. 2), and that actions unsuitable for automation, i.e. Gathering-information, Deciding-deal-type have been omitted. Although allocation of actions to people or computers uses general human factors heuristics, JSD actions can give some guidance. The actions which conformed well to the JSD concept were more suitable for computerisation, whereas actions with less precise boundaries suggested allocation to people. Other actions were added as a consequence of the cognitive dimension in task analysis.

3.2. Task complexity analysis

The second addition of JSD was a set of simple metrics for estimating the potential cognitive problems in user-system interaction. Following production rule approaches to complexity,[15] task actions were counted and conditional statements scored for the complexity of their logic. This gave an overall metric for each PSD diagram, as illustrated in Table 1 for the *Make-Deal* task.

Table 1. Calculation of task complexity for the Make Deal task

Component	Average	Range
Actions	9	8–10
Simple conditions	5	4–6
Complex conditions	1	—
Overall complexity*	16	14–18

* Complexity = actions + simple conditions + (complex conditions × 2).

Not all actions or conditions would be invoked in each task execution (e.g. trying the broker was often not necessary) hence a range of actions and conditions is given. The metric does not allow for the number of iterations for an action as this is difficult to estimate. All the conditions were simple apart from Comparing Risks.

The complexity metric was used for several purposes: to indicate the need to decompose complex tasks; to allocate tasks to people matching user skills to task complexity; and to provide variety in an individual user's work by giving a mixture of tasks at different complexities. Given the small number of user roles analysed in the case

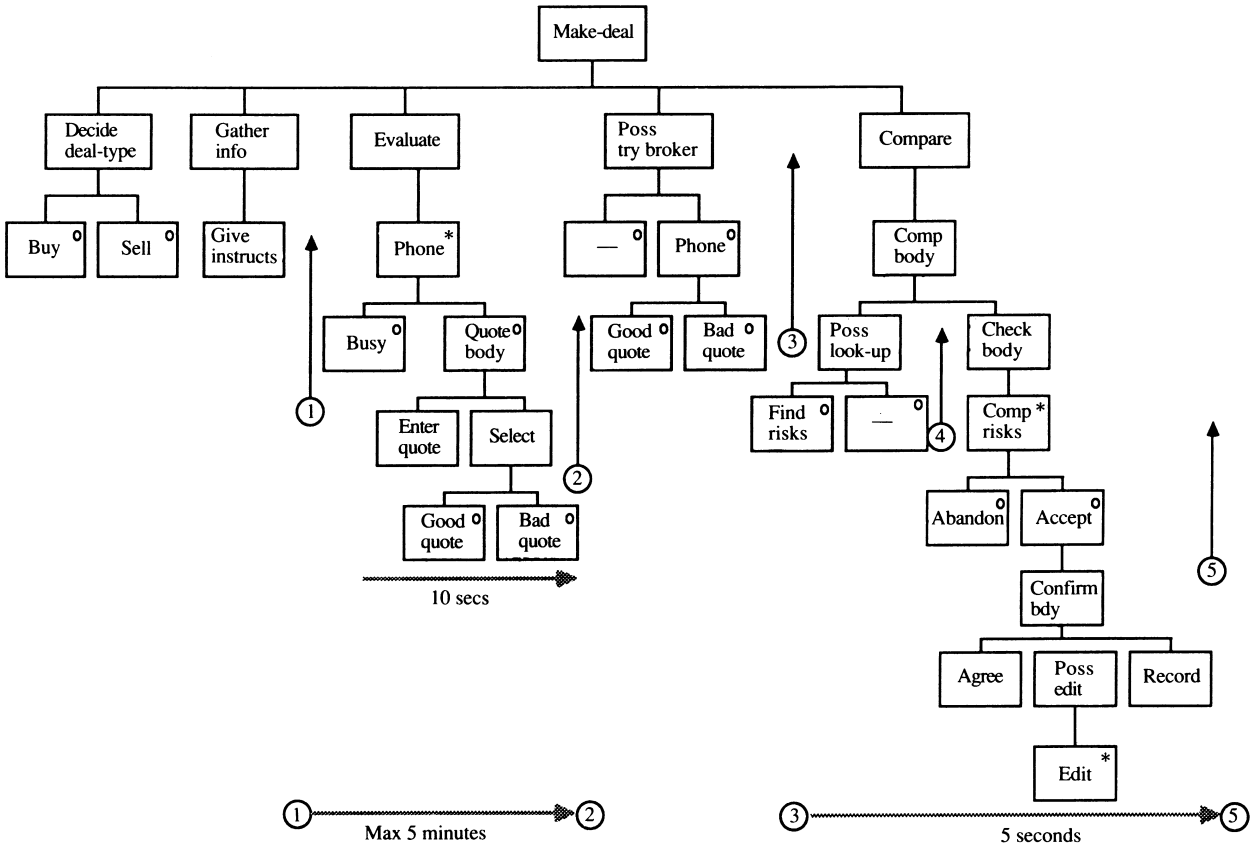

Figure 5. Timing and breakpoints – user task description: make deal.

study (Dealer and Head dealer) it was difficult to take much initiative in task/work design. However, where possible, less complex tasks were interspersed with more complicated ones.

3.3. Timing analysis

PSDs were used to specify the timing constraints imposed on task operation. This facilitated review of task descriptions to ensure that people can carry out a task in the necessary time, and to plan computer response times that are sufficiently fast for effective operation. Subsequences of actions were marked with timing constraints as illustrated on the PSD in Fig. 5, to answer the question 'when must these actions be done'. The task specifications were examined to determine whether it would be possible for the human operator to complete certain actions in specified time. The same diagrams were also used for computer system response time planning to ensure the user-system dialogue allowed task to be completed within the time constraints.

Timing analysis prompted further specification in several parts of the dealing system. For instance, in the evaluate deal task entering and selecting quotations had to be achieved within 10 seconds. It was doubtful whether this would be possible using a conventional keyboard for data entry. Consequently voice data entry and special dedicated function keys were investigated. It was determined that dealers could reasonably be expected to perform only 4–5 keystrokes per quotation. The design solution was to have broker-IDs on dedicated keys and to require the dealers to enter only the last few digits of the rate. Rates for foreign-exchange quotations usually differ in least significant decimal places, e.g. £1 = 9.8275 or 9.8281 French francs.

The task-action sequences were analysed for overall length. Long sequences of actions can cause fatigue, loss of concentration and stress. It is important to design natural pauses in tasks to allow users to do a 'mental reset', thereby forestalling loss of attention and fatigue. Reset points, alias closure events, were planned for long, continuous task by inserting breakpoints into long sequences. This analysis was used to plan response times and screen messaging to encourage closure events in dialogues. Longer response times (2–4 seconds) are appropriate for breakpoints, especially at the end of groups of logically related actions. While the positioning of breakpoints is based on the analyst's knowledge of the task domain, the JSD specification can help as breakpoints as indicated at the end of sub tree sequences.

In the *Evaluate-Deal* task (see Fig. 5) breakpoints would naturally be suggested at the end of the Evaluate sub tree. However, the overall timing constraints on the dealing task meant that longer response times could not be used. In this case the realities of the dealer's hectic and stressful job could not be avoided. Representation of timing constraints on the main specification document helped investigation of these issues.

3.4. Information analysis

Whereas task specifications can provide the basis for design of the user-system dialogue, another starting point is required for design of interface displays. This involves analysis of the information requirements necessary for successful operation of a task.

Specification of the data necessary for actions is part of the JSD method in which input messages are described as 'action attributes'. JSD also recognises status and descriptive data, belonging to a process, as 'entity attributes'. JSD function processes access entity attributes (or state vectors) belonging the entities in the model sub-system. State Vector accesses are discovered by asking the question 'what information is required at this step in a task?'. The information requirements are found in the definition of JSD entity attributes for SV access and in the attributes for each action.

Action attributes contain much redundancy as data items are often shared between several actions. The initial action/attribute lists were slimmed down into information groups by linking logically related data items (see Table 2). These groups are frequently associated with an entity or an input transaction. Standard practices of data analysis, e.g. entity, attribute, relationship modelling can help here. In the dealing system the information groupings for the *Evaluate-Deal* task were related to entities such as Counterparty, and the FX transaction. The users were consulted about their views on display requirements which elicited other information groupings not directly related to entities or transactions (e.g. Position, Option-list, Risks – which are attributes of a counterparty entity). Generally, HCI advises taking the user's model of information and tasks,[25,27] so the users' suggestions were incorporated into the design.

Table 2. Task design: information requirements

Number	Action	Attributes
		Action attributes
1.	Decide deal	Trans type
2.	Gather-info	Currency type, duration, amount, trans-type
3.	Phone	Counterparty-code, phone number
4.	Quote	Rate, duration, curr-type
5.	Calculate	Rate, duration, amount, yield
6.	Select (broker)	Counter-party, yield Broker ID
7.	Find risks	Counterparty, Econ risk, political risk
8.	Compare	Counterparty, yield, risks
9.	Add sum	Currency-type amount, duration

Main group	Sub group
	Information display units
1. FX transaction	
Currency type	
Amount	
Duration	
Trans-type	
2. Counterparty	Counter-part-reply
Counterparty-code	Quoted-rate
Phone number	Duration, yield
3. Option-list	
Counterparty-code	
Yield	
4. Broker-list	
Broker-ID	
Yield	
5. Risks	
Political risks	
Economic risks	
6. Position	
Currency amount, duration	

The action sequence was traced to specify which information groups were required for each action in a task. The result was recorded in a bar chart format as illustrated in Fig. 6, to show when and for how long information should be displayed in relation to the action sequence actions in a task. Most information groups were only required by one or two contiguous actions, hence the display order approximately matched the action sequence (see Fig. 6). Transaction-related information was required for nearly every action so this data was preserved on the screen throughout the task. The task-information specification was transformed into display screens, windows and messages, etc. embedded within a dialogue sequence. Information analysis allowed the screen display sequence to be planned so that display elementary operations could be added to the appropriate PSD actions, and ensuring that the necessary information was preserved on-screen once it has been displayed.

A further refinement of this analysis was to calculate the loading on the user's working memory at each task step. A general human factors heuristic is that people should not be required to memorise too many facts which are necessary for completion of a task.[22] It is necessary to decide how much information has to be displayed on the screen and how much could possibly be held in the user's memory. The JSD concepts of action and entity attributes can be generalised in this context to information held in working and long-term memory respectively. To evaluate memory loading, tasks were examined to describe the information necessary for human decision making. Approximate quantification of memory loading was carried out by recording data items by size, complexity of data structure and probable familiarity of the item to the user.

If either the quantity of data exceeded working memory using the simple, seven plus or minus two metric,[17] or most of the data was expected to be unfamiliar to the user, then this suggested that users' memory should be supported with computerised information displays. When this metric predicts excessive memory loading, further consultation with user is indicated to investigate possible computerised support. While it is impossible to give accurate quantification of memory loading based on simple counts of data items;[9] this approach does allow quick and simple estimates of possible memory loads to help design of task support displays.

Memory loading analysis of the Compare step in the *Make-Deal* task is illustrated in Table 3. Three groups of information were required, the FX transaction, counterparty quotes on the option list and risks. Even assuming the users were familiar with the data, which reduces the memory load; the minimum memory loading was 8 units which exceeded the limit. The loading becomes progressively worse with a longer option list. Furthermore this may be compounded by the need to hold partial results in memory (e.g. that the first three counterparties were bad while considering the next one). Further consultation

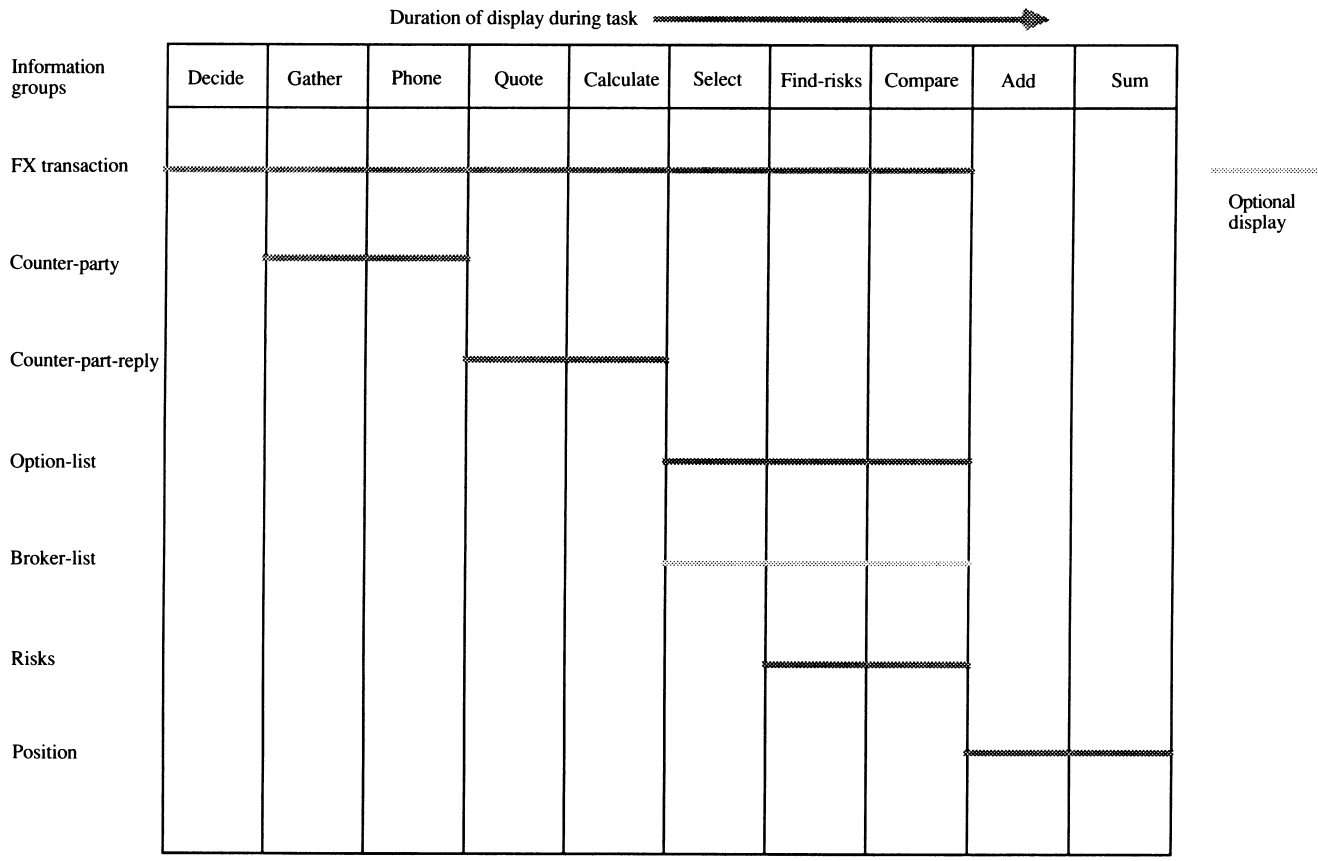

Figure 6. Display bar chart.

Table 3. Working memory analysis

Task: Make Deal

Action/Decision: Compare risks

Info Groups	Min	Max units	Familiar?	Notes
FX Transaction				
Currency type	1		Y	Set code
Amount	5	9	N	
Duration	1	1	Y	Set days
Trans type	1	1	Y	Buy-sell
Option List				
Counterparty	1	3	Y	Max 3/list
Yield	3–6	9–18	N	*
Risks				
Political	1	3	Y	Set codes
Economic	1	3	Y	†
Total units	14	38		

* Value of yield may be variable.

† Assumes risks aligned with counter party IDs. Risk code may not be present for all counterparties.

Notes: Information groups are taken from Table 2. Minimum working memory requirements assuming only 1 counterparty on the option list and maximum familiarity with items exceeds the limit. Information support displays are suggested for all three groups, with counterparty information aligned to reduce the need for searching.

with the users confirmed the need for information support displays. In addition, this analysis also focused attention on how the information was used, e.g. in the comparison counterparty quotes are weighed up against the risks. This suggested that this information needed to be placed together and aligned for quick visual scanning.

4. TRANSFORMATION TO DIALOGUE DESIGN

First, task specifications were elaborated to add actions necessary for operation of the human computer interface. JSD specifies the input subsystem with function processes. There are two types of functions called filter processes. Context filters deal with validating event input with reference to the life histories of entities; whereas simple filters implement the user interface and other forms of validation. The task design proposed in this study essentially elaborated the simple filter. The task design represents a model of what the computer should do to help the user achieve a job of work. Actions to control the computer operation have to be added to the PSDs. This is effected by modelling the user's operation of the interface by asking the question 'what might the user want to do at this point in the dialogue?'.

HCI guidelines are used to inform the answer.[24,27] The PSDs were elaborated to add validation checks on input and user control actions such as:

Escape – might the user want to exit from this operation?
Help – information to guide the user at this step.
Undo – go back to a previous state undoing recent processing, etc....

Because much user interaction is uncertain the JSD design construct of backtracking was used to deal with

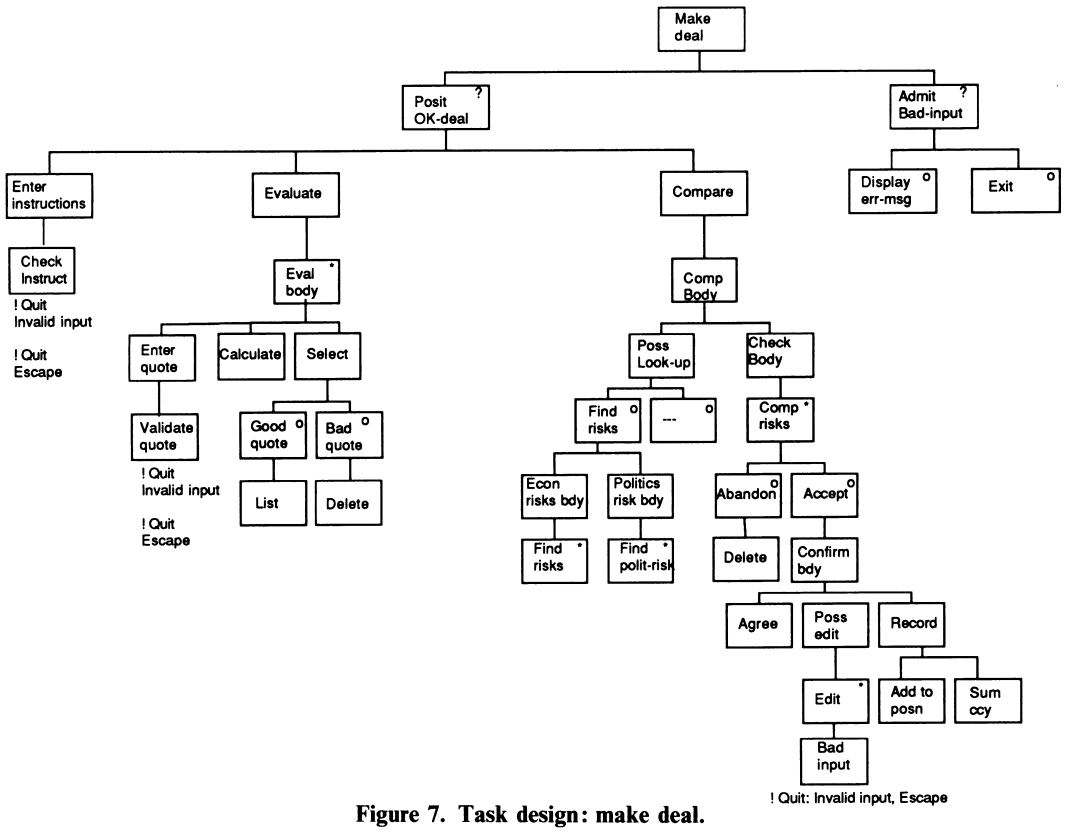

Figure 7. Task design: make deal.

run time uncertainty e.g. the indeterminate arrival of invalid data and escape commands.

The next step was to add display requirements using the information analysis bar chart (Fig. 6). Display instructions were included as elementary operations and assigned to PSD components. For this purpose the PSD is not ideal because, although it shows when information is displayed, it does not show for how long it should be displayed. Display requirements were taken from the display bar chart and the description of information groups. Then user support displays were added, following HCI guidelines.[24, 27] For example:

* User feedback messages: add messages after input actions to acknowledge data entry and commands.
* User guidance: add messages before input actions to inform user of expected actions or options available
* Error messaging: explicit and informative messages should be added on invalid actions
* User guidance-status information: messages should be added at the start of task sequences and at the boundary of sub-tasks to inform the user of any changes in system status.

In this way PSDs were elaborated into program specifications by addition of further design detail. An alternative method of design, or a valuable cross check, is to diagram the user-system dialogue as a transition network diagram. The nodes correspond to display states and arcs match to actions (e.g. transition between states). The Jackson process structure diagram is formally equivalent to a state transition diagram even thought it records events rather than states. Dialogue network (as state transition) diagrams make tracing of error and escape pathways easier; however, they are not so suitable for display specification, especially if several messages have to be displayed concurrently over long time periods.

A filter process PSD for the *Make-Deal* task is illustrated in Fig. 7. Quit operations allow the user to escape from an option if it has been entered by mistake. Other quits trap validation errors and switch control to the Admit Bad input part of the process. At this stage specification of the dialogue and display was combined on one PSD diagram by the addition of elementary operations for display, data entry, and similar design details.

Elementary operations for dialogue and display control have been included on the diagram, although some have been omitted for sake of brevity and simplicity of the specification. A particular case in point is the Help system which has to be available at nearly every step.

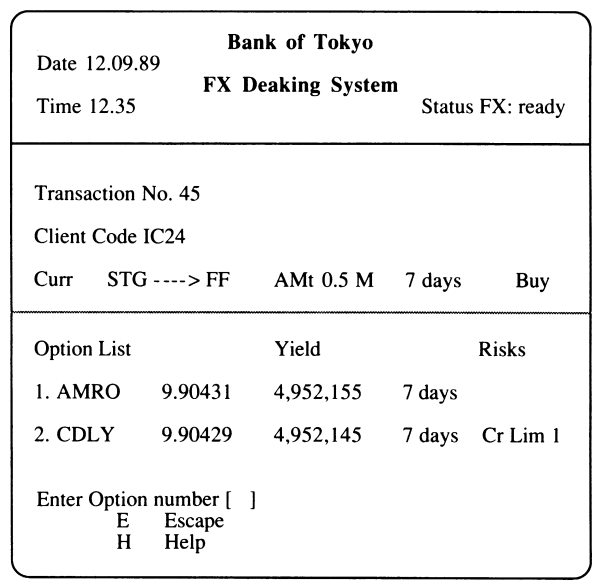

Figure 8. Prototype dealing screen.

This requires pseudo-concurrent specification which, although possible in JSD notation, is not very tractable.

At this stage two routes are possible. Either the specification could be coded in a third generation language using the detailed specification which JSD produces or the user interface could be prototyped using 4th-generation tools. The latter course was chosen as a better way of validating the design before expensive final implementation decisions were made. Prototype displays were designed for the *Make-Deal* task as shown in Fig. 8. The main transaction details are placed at the top and held there throughout the task. The deal options and risks are aligned for easy scanning.

This work is still in progress, although preliminary results suggest that the designs are sound and that most problems have been anticipated by the JSD-HCI analysis.

5. EVALUATION OF THE JSD-HCI METHOD

This section reports experience in using the method on the dealing system application.

The method was easy to use even though it was used by a developer with no formal HCI training. Task analysis using the PSD notation worked well and focused specification on the users' needs. Task complexity analysis was useful within the limitation that the length of tasks varies according to how many actions are executed in iterations and selections. It proved helpful as a rough guide when allocating tasks to users. The PSDs did help planning breakpoints in tasks which were manifest later in the design of response times.

Information analysis helped specification of interface displays; although, the memory loading analysis proved to be problematic. The procedures and guidelines for calculating memory loading were unsatisfactory because it was hard to discover users' familiarity with data items. Furthermore, the metric is open to criticism because actual storage of information in working memory is not a simple matter of units. In hindsight this analysis may not have paid off in improved quality of design, although it did focus attention on important HCI issues.

Overall it was estimated that addition of the HCI components added 20–30% to the specification effort compared with using JSD alone. It was felt that this effort was well worth while in creating a better-quality design.

6. DISCUSSION

Solving the complex human factors issues raised in systems development requires extensive HF skills training; however, experiences reported in this study suggest that improvements in HCI practice can be delivered by HCI-software engineering method integration. Ideally, HCI design should be supported by intelligent human factors CASE tools. Prototype tools incorporating HCI guidelines have been produced[36] but their value has yet to be proven and the application of guidelines in correct contexts is known to be difficult.[27] Furthermore, these tools are not integrated with CASE environments, so industrial practice is doubtful.

A more profitable approach may be expert systems to support HCI design;[35] although considerable theoretical development is required before practical products can be delivered. In the meantime, HCI-software engineering method integration is necessary to plug the human factors expertise gap. It is notable that movements are being made in this direction both by HCI researchers who are making methods more relevant to system development practices[7,13,33] and authors of commercial structured system development methods who are beginning to address the issue of integrated HCI-SE development.[16,30]

In spite of these advances, much still needs to be done. The majority of software engineering methods pay no attention to the cognitive aspects of designing interfaces for people. Indeed, most software engineering methods do not consider user centred issues (see Olle *et al.*[20,21] for surveys of method qualities). On the other hand, HCI methods are driven by a variety of analytic and evaluative motivations (see a review by Simon[23]) and none cover all the analysis and design issues in systems development.[28] Furthermore their track record of practice is poor.[4]

Only a few HCI methods have approached the integration problem. Of these Task Knowledge Structures (TKS[14]) presents a comprehensive task analysis with procedures for transforming the task knowledge specification into a software design. At present, TKS specifies the procedural, structural and planning knowledge about tasks and *inter alia* systems. The design/implementation transformations of TKS are oriented towards knowledge based languages (e.g. frames, Lisp/LOOPs) which limits its applicability to current commercial practice, i.e. methods such as SSADM and JSD, and procedural languages.

The USE method of Wasserman[32] is closer to standard software development practices, being based on data flow diagramming techniques derived from Structured analysis.[6] Unfortunately USE delivers little cognitive input into design of interface software and relies on a dual approach of data flow diagrams and 'outside in' development of dialogues with transition network diagrams.[33] The harmonisation of these techniques and their relative contribution to a design is not clear.

If integration is to be approached from the software engineering side, Jackson System Development,[10,26] has many attractive qualities. First, the PSD notation of JSD is particularly suited to the state/event nature of dialogue design and provides a clear description of tasks. Process structure diagrams are appealing in providing a consistent notation for task analysis and dialogue design. With suitable annotations the same diagram can be used for several aspects of HCI specification. This has the virtue of enhancing consistency throughout the specification by creating a diagrammatic lingua franca. This approach also links dialogue and display design, creating an integrated interface specification not found in any previous HCI or SE method.

JSD has been augmented by a separate method for HCI specification – ATOM Analysis for Task Object Modelling.[31] This approach still maintains the traditional separation of HCI-SE specification which may well militate against its acceptance by software engineers. A further criticism of ATOM is that it adds little in the way of a cognitive task analysis. The utility of this method has yet to be proven.

While specification of dialogues by JSD may not rival the comprehensivity of languages designed for that purpose,[11] there is evidence that PSDs are a more

tractable notation for software developers than state transition diagrams and other formal HCI notations in CLG[18] and CCT.[4] Although the PSD notation is effective for describing sequences of interaction, specification of concurrent events is often necessary. JSD can show concurrency in the SSD notation, but this is not adequate for specifying synchronisation of dialogue events. This will be necessary for specification of complex multi-tasking interfaces in which concurrent events may take place in two or more windows.

Although software engineering methods can address task specification and design of interactive software, there are issues which in their present form, they can not. First, is cognitive task analysis to ensure design takes account of the limitations in human information processing. The simple approach presented in this study may be justified in cost/benefit terms; although more research is required to establish the trade off between the costs of more sophisticated cognitive analyses, and the consequent improvements in design quality. Synthesis of HCI techniques within SE methods may improve usability in software designs. In requirements engineering and cognitive analysis, method symbiosis may be more advantageous.[28]

Analysis of human activity, i.e. user characteristics-task matching, and specification of jobs, all require addition of HCI methods and techniques to JSD, as noted by Walsh *et al.*[29] Methods exist to address these concerns, for example USTM,[8] SSM,[5] and see Sutcliffe[28] for further detail. Inter-method cooperation is a realistic means of encompassing the range of requirements elicitation, task analysis and organisational issues, considering that single methods are usually targeted on a small number of well defined issues.

The challenge for specialists in Human Computer Interaction is to deliver practical, useful techniques which will help software engineers produce more usable products. It is contended that the approach of incorporating HCI procedures and practice with structured system development methods is essential to improve the usability of software, given the low acceptance of specification and design methods created within the HCI community. While the theoretical validity of integrated HCI-SE methods, and indeed structured methods themselves, remains to be proven; method synthesis followed by evaluation in commercial practice is essential to make progress. Accordingly, the proposals in this study are being subjected to further tests of industrial practice.

Acknowledgements

The authors are indebted to helpful suggestions on parts of this work from Paul Walsh, and Keong Lim at the Ergonomics Unit, University College, London.

REFERENCES

1. R. W. Bailey, *Human Performance Engineering: A Guide for System Designers*. Prentice Hall, Engelwood-Cliffs (1982).
2. P. Barnard, Cognitive resources and the learning of human computer dialogues. IBM Hursley Research Centre Report No. HF 118 (1986).
3. V. Bellotti, Implications of current design practice for the use of HCI techniques. In *People and Computers*, vol. IV, edited D. M. Jones and R. Winder, pp. 13–34. Cambridge University Press (1988).
4. V. Bellotti, A framework for assessing applicability of HCI techniques. In *Proceedings INTERACT – 90*, edited D. Diaper, D. Gilmore, G. Cockton and B. Shackel. North Holland, Amsterdam (1990).
5. P. Checkland, *Systems Thinking, Systems Practice*. J. Wiley, London (1981).
6. T. De Marco, *Structured Analysis and System Specification*. Yourdon Press, New York. (1978).
7. L. Damodaran, K. Ip and M. Beck, Integrating human factors principles into structured design methodology: a case study in the U.K. civil service. In *Information Technology for organisational systems*, edited H. J. Bullinger *et al.*, pp. 235–241. Elsevier (1990).
8. C. Fowler, M. Kirby, L. Macaulay and A. Hutt, User skills and task match (USTM): a human factors based methodology for determining product requirements. In *Proceedings of the 4th Alvey Conference*. Swansea, Wales (1988).
9. G. J. Hitch, Working memory. In *Applying Cognitive Psychology to User Interface Design*, edited B. Christie and M. Gardiner. J. Wiley, London (1987).
10. M. A. Jackson, *System Development*. Prentice Hall, Englewood Cliffs, NJ (1983).
11. R. J. K. Jacob, A specification language for direct manipulation user interfaces. *ACM Transactions on Graphics* **5** (4), 283–317 (1987).
12. P. Johnson, Towards a task model of messaging: an example of the application of TAKD to user interface design. In *People and Computers: Designing the Interface*, edited P. Johnson and S. Cook, pp. 46–62. Cambridge University Press (1985).
13. H. Johnson and P. Johnson, The development of task analysis as a design tool. A method for carrying out task analysis. Internal Report, Dept of Computer Science, Queen Mary College, University of London (1987).
14. P. Johnson, H. Johnson, R. Waddington and A. Shouls, Task related knowledge structures: analysis, modelling and application. In *People and Computers*, vol. IV, edited D. M. Jones and R. Winder, pp. 35–62. Cambridge, Cambridge University Press (1988).
15. D. E. Kieras and D. Polson, An approach to the formal analysis of user complexity. *International Journal of Man Machine Studies* **22**, 365–395 (1985).
16. Learmont and Burchett Management Systems, *LBMS Structured Systems Development Methodology*, Version 3. LBMS, London (1986).
17. G. A. Miller, The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review* **63**, 81–97 (1956).
18. T. P. Moran, The Command Language Grammar: a representation for the user interface of interactive computer systems. *International Journal of Man Machine Studies* **15**, 3–50 (1981).
19. D. G. Morgan, D. N. Shorter and M. Tainsh, Systems engineering; a strategy for the improved design and construction of complex IT systems. DTI, London (1988).
20. T. W. Olle, H. G. Sol and A. A. Verrijn-Stewart (eds), *Information Systems Design Methodologies: Improving the Practice*. North Holland, Amsterdam (1986).
21. T. W. Olle, J. Hagelstein, I. G. Macdonald, C. Rolland, H. G. Sol, F. J. M. Van Assche and A. A. Verrijn-Stuart, *Information Systems Methodologies: A Framework for Understanding*. Addison-Wesley, Reading, MA (1988).
22. B. Christie and M. Gardiner, (Eds), Applying Cognitive Psychology to User Interface Design. J. Wiley, London (1987).

23. T. Simon, Analysing the scope of cognitive models in human computer interaction: a trade-off approach. In *People and Computers*, vol. IV (HCI – 88), edited D. M. Jones and R. Winder, pp. 79–96. Cambridge University Press (1988).
24. S. Smith and J. N. Mosier, Design Guidelines for User-System Interface Software. Mitre Corp, Bedford, MA (1986).
25. B. Shneiderman, *Designing the User Interface*. Addison Wesley, Reading, MA (1987).
26. A. G. Sutcliffe, *Jackson System Development*. Prentice Hall, London (1988).
27. A. G. Sutcliffe, *Human Computer Interface Design*. Macmillan, London (1988).
28. A. G. Sutcliffe, Task analysis, systems analysis and design: symbiosis or synthesis? *Interacting with Computers* **1**, 6–12 (1989).
29. P. Walsh, M. K. Carver, K. Y. Lim and J. B. Long, An approach to specifying the user interface using human computer interaction in conjunction with Jackson System Development. Ergonomics Unit Report, University College, London. To appear in *Software Engineering*, edited J.Hemsley. Pergamon Infotech, Oxford (1988).
30. P. Walsh, K. Y. Lim, J. B. Long and M. K. Carver, Integrating human factors with system development. Ergonomics Unit Report, University College, London. In *Designing End-User Interfaces*, edited N. Heaton and M. Sinclair. Pergamon Infotech, Oxford (1988).
31. P. Walsh, Analysis for Task Object Modelling (ATOM): towards a method of integrating task analysis with Jackson System Development for user interface software. In *Task Analysis for Human Computer Interaction*, edited D. Diaper, pp. 186–209. Ellis Horwood (1989).
32. A. I. Wasserman, Developing interactive systems with the user software engineering methodology. In *Proceedings Interact – 84*, vol. 1, edited B. Shackel, pp. 471–477. North Holland, Amsterdam (1984).
33. A. I. Wasserman, P. A. Pircher, D. T. Shewmake and M. L. Kersten, Developing interactive information system with the User Software Engineering methodology. In *Readings in Human Computer Interaction: A Multi-disciplinary Approach*,edited R. M. Baecker and W. A. S. Buxton. Morgan Kaufman, Los Altos, CA (1987).
34. A. Whitefield, Models in human computer interaction: a classification with special reference to their uses in design. In *Interact – 87*, edited H.-J. Bullinger and B. Shackel, pp. 57–64. North Holland, Amsterdam (1987).
35. M. D. Wilson, P. J. Barnard and A. Maclean, Task analysis in human computer interaction. IBM Hursley Research Centre Report No. HF 122 (1986).
36. P. Pettitt, INTUIT, A knowledge and structured design approach to user-centred design. In Proceedings of Esprit Technical Week '89, pp. 903–914. Kluwer, Dordrecht (1989).

Announcement

2–6 SEPTEMBER 1991

Eurographics '91, Annual Conference of the European Association for Computer Graphics, Hofburg, Vienna, Austria

Call for participation

Eurographics '91 is the twelfth annual event of the Eurographics Association. It is the leading international computer graphics conference in Europe, and a vital meeting point for researchers, practitioners, teachers, vendors and users. It is a forum for the latest developments in graphics technology, for case studies in graphics systems and applications, and for surveying the state of the art. The location for this event is Vienna, Austria, the door to Eastern Europe, whence many participants and visitors are expected. One year before the establishment of the Common European Market this is your great chance to get an overview of what's going on in the field of computer graphics in Europe.

Themes of the conference include: Image Synthesis and Animation, Scientific Visualisation, Graphics Hardware, Modelling, Image Processing, Human–Computer Interaction and Graphics Standards and their applications.

Tutorials, 2–3 September 1991

The first two days of the event will be devoted to the tutorial programme. Tutorials will be given by leading international experts and will cover a wide range of topics offering an excellent opportunity for professional development in computer graphics and related areas.

Technical programme, 4–6 September 1991

Papers have been invited in computer graphics, which are either research contributions (original and recent developments) or practice and experience papers (case studies in system development, or applications).

Conference proceedings will be published by North-Holland, and will be available at the conference. From the papers selected for presentation, an international jury will select the best paper and the author will receive the Günther Enderle Award.

State-of-the-art reports

State-of-the-art reports will be selected by the International Programme Committee and published.

Poster presentation

In addition to the lectures, Eurographics '91 will organise a poster presentation. This offers the opportunity to display material on boards in a special area. The poster presentation aims at publicising work in progress, results achieved after the deadline for submission of papers, results which are less appropriate for oral presentation to a larger audience, etc.

Exhibition, 4–6 September 1991

A major industrial exhibition on computer graphics and applications will be held in conjunction with the conference. This event will be the largest event of this kind in the field in Vienna. Although it is fully integrated with the conference, it will be open to the public. The latest achievements in computer graphics hardware, software and applications will be presented by a wide variety of international and local companies. Guidelines for exhibitors can be ordered from the Conference Secretariat.

Slide, video and film competitions

There will be a competition of computer-generated videos, films and 35 mm slides, with prizes awarded for the best entries based on creativity and technical excellence. A broad selection of all material entered will be shown during the conference in a video show also open to the public.

Graphics R&D in European Community programmes

Eurographics '91 wants to serve as a forum for the presentation and discussion of computer graphics-related results of R&D programmes financed by the EC. In three sessions R&D project work within EC programmes will be presented.

Vienna

Visit Vienna, the European capital of music, in 1991, the year of the 200th anniversary of Mozart's death! You will be fascinated by the conference centre, the former palace of the Austro-Hungarian emperor. You will love the city with its beautiful buildings and the wonderful surroundings.

There are substantial discounts on the conference fees for Eurographics members. For details of EG membership, please contact Eurographics Association (CGW), P.O. Box 16, CH-1288 Aire-la-Ville, Switzerland.

For further information please contact:

Eurographics '91 Conference Secretariat, c/o Interconvention, Austria Centre Vienna, A-1450 Vienna, Austria. Tel: +43 (1) 2369 2640. Fax: +43 (1) 2369 648.