Designing Complex Systems - a Structured Activity

Gerrit C. van der Veer (+, ++), Johannes C. van Vliet (+), Bert F. Lenting (++)

(+) Dept. of Computer Science, Vrije Universiteit, Amsterdam, NL
(++) Dept. Ergonomics, CTIT, Twente University, Enschede, NL

ABSTRACT

This paper concerns the development of complex systems from the point of view of design as a structure of activities, related both to the clients and the users. Several modeling approaches will be adopted for different aspects of design, and several views on design will be integrated. The proposed activity structure is based on teaching design practice, and will be illustrated by examples from design courses for university students and for practitioners in industry.

1. COMPLEX SYSTEMS AS A CHALLENGE FOR DESIGN

This contribution focusses on the design of that part of information systems, that is facing the users. We will only consider those aspects of the system that are of relevance to the user in working with the system. In terms of the Seeheim model (Pfaff and ten Hagen, 1985) these are the parts indicated by the user interface and the application interface. Tauber (1988) and Van der Veer et al. (1985) indicate this as the UVM, the user's virtual machine, denoting all aspects of a system the user should be aware of during interaction (including planning and evaluation of interaction).

Traditional user interface design mainly concerns the situation of a single user and a monolithic system. In current applications of information technology the UVM should include all aspects of communication between the user and other users of the system as far as communication is routed through the system. It should also include aspects of distributed computing and networking as far as this is relevant for the user, like the access to and structural and time aspects of remote sources of data and computing.

The newer types of application bring another dimension of complexity into view. People are collaborating in various ways mediated by information technology. Collaboration via systems requires special aspects of functionality, both providing facilities for the integration of actions originating from different users on shared objects and environments, providing facilities to manage and coordinate, and providing communication functionality.

The concept of "user interface" in the literature is used to denote various parts of the UVM. Unless otherwise indicated, in this paper it will indicate the whole of the UVM, including user relevant aspects of communication and network structure, and of collaboration. As far as these extensions are an important aspect of a system, current literature often denotes such systems by the label "groupware".

Our approach does not bother with this distinction. It is claimed to be relevant for both the situation of the classical single user systems and the groupware case. In fact we expect this distinction will no longer be important or even valid in the near future.

1.1 The task world

From the point of view of the user interface, designing systems for users means designing a task world in the first place. In fact, there is already a task world to start with in most cases, which may not employ information technology, or use systems that should be improved. Consequently, design will start with knowledge acquisition about the current task world and redesigning the task model in relation to identified problems and requirements stated by the principal of the system.

We need to collect knowledge about the task domain, to model it and to analyse it. As our design approach covers groupware as well as single user situations, we need a structural framework that covers both. Jordan (1994) analyses the types of task knowledge needed for systems design along two dimensions: individual vs group knowledge, and explicit vs implicit knowledge. The combination of both dimensions indicates the type of knowledge that has to be collected and hints to methods that can be applied for identifying and structuring this knowledge. For the explicit knowledge the methods in general amount to the straightforward acquisition of knowledge and skills (psychological methods combined with behavior observation) in the individual case, and the collection of documents and archival search in the group case. For the implicit knowledge the methods might be more complicated and might need a careful methodology in order to guard reliability. Observation and hermeneutic methods will be combined in the individual case, and ethnographic methods will be needed for the group situation.

Most task analysis methods are Taylorean in nature: tasks are recursively decomposed into simpler tasks and each task has "one best way" to accomplish it. By careful observations and experiments this one best way can be found and next implemented. Hirschheim and Klein (1989) stress that the Taylorean, functional approach is not the only, and need not be the most appropriate, approach to task analysis. They derive four paradigms for information system development. These are related to two dimensions of task knowledge (world knowledge in their terminology): subjectivist-objectivist and order-conflict. The subjectivist-objectivist dimension is related to both of Jordan's dimensions, though they do not completely overlap. The conflict aspect is not explicitly covered by Jordan's framework, though it certainly is an aspect both of the task knowledge in groupware situations, and of the differences in task knowledge between end-users and between end-users and their principals. The order-conflict dimension of Hirschheim and Klein stresses the fact that the analyst is not a passive outside observer of the Universe of Discourse, but that he actively participates in shaping it. In our view, this holds the more for groupware applications.

1.2 The functional world

Having identified suitable methods for knowledge acquisition, we need a conceptual framework to model task knowledge. Van der Veer (1994) derives a structure of concepts to model the functionality of complex systems and groupware. The framework considers 3 viewpoints: (a) people, (b) work, and (c) situation.

a. Modeling people is based on specifying types of roles and their task-related attributes, specifying an actor type and its attributes (actors are individual users), and specifying the organisation, i.e., the relations between actors and roles in the task domain. The specification of these people related concepts will include relations to the concepts that are specified from the other two viewpoints.

b. Modeling work means specifying tasks and a task structure, actions that relate to tasks, and protocols and strategies (to be identified in the case of real world situations) or procedures (specified when task structures are designed) to indicate action structures related to situations and roles. Each of these types of specifications is done by coupling relations to the work concepts, and to the concepts regarding people and situations.

c. Task delegation to information systems, from the users' points of view, should be considered situated", i.e., the situation has to be modeled and designed as part of designing the user interface and functionality. Modeling the situation means specifying object types (in the meaning of "things" people manipulate in the course of performing or delegating tasks), the structure of objects (both the type hierarchy and the semantic relations between objects), and specifying the situation where certain tasks are performed, being a space where people act in relation to each other and on objects.

1.3 The representations

One of the viewpoints on design is that of developing representations. Three classes of representations have to be developed: (a) users of a system have to communicate their intentions, (b) systems have to represent information to users, and (c) designers have to represent their design decisions and solutions.

a. *The users' representations of their intentions towards the system.* This class of representations will be refered to as the users' language. Designing a user interface includes designing the users' language. Users will have to be able to express their task delegation, to answer system questions, and to ask for information about various aspects of interaction (like the semantics, syntax and lexicon of the interaction language, the state of the system and network, the history of transactions). This part of the design process needs a framework, in order to cope with the diversity of design decisions and the complexity of the interrelations. Classical HCI has developed a variety of analytical methods and modeling techniques for this purpose. Based on comparative studies (de Haan et al, 1991), we consider methods like ETAG (Tauber, 1990) to be an adequate basis. It provides a well structured formalism, based on both psychological relevant semantic concepts of the users' language and on object oriented design methods that enable application of state of the art software engineering techniques. Figure 1 shows some fragments of an ETAG specification that was developed in one of our design classes. It specifies some conceptual objects and events of a calendar system. The example also shows one of the "basic tasks", equivalent to a single command or task delegation to the computer system, and it defines the users' language towards the system via production rules.

CONCEPTUAL OBJECTS

type [OBJECT > TIMESLOT]

themes: [MEETING: {*m}@0-1];
relations: [place.IN([CALENDER])];

end [TIMESLOT].

type [OBJECT > MEETING]

themes: [PARTICIPATION_LIST];
relations: [place.IN([TIMESLOT])];
attribute: <KIND>;

end [MEETING].

type <ATTRIBUTE > KIND_OF_MEETING>

object type: [MEETING]
value set: {"group","personal","work commitment"}

end <KIND_OF_MEETING>.

CONCEPTUAL EVENTS

type [EVENT > SEND CANCEL_MESSAGE]

description: for [TIMESLOT: *t]
[event.CREATE ([CANCEL_MESSAGE],[TIMESLOT])]
[event.SEND([CANCEL MESSAGE],
[PARTICIPATION_LIST])]

precondition: [state.FILLED([TIMESLOT: *t])]

comment: "create cancellation message for timeslot t and send it to all participants."

end [SEND CANCEL_MESSAGE].

type [EVENT > CANCEL MEETING]

description: for [TIMESLOT: *t]
[event.SEND CANCEL_MESSAGE([TIMESLOT])]
[event.COPY MEETING TO TODO_LIST
[event.EMPTY TIMESLOT([TIMESLOT])]

precondition: [state.FILLED([TIMESLOT: *t])]

comment: "cancel meeting in timeslot t"

end [CANCEL MEETING].

Figure 1.a: Example ETAG specification (fragment)

Like all methods in this category, ETAG has its restrictions. Task level knowledge (Moran, 1981) is not modeled in the "standard" ETAG technique, so we have to rely on extensions for this purpose. Thus far we developed an object oriented task modeling approach (van der Veer, 1994) that is based on MAD (Sebillotte, 1988; Scapin and Pierret-Goldbreich, 1989), Johnson's TKS (1989), and on Jordan's (1994) structured ethnographic task description methods.

b. *The system's representations for the user*. The system has to represent information on behalf of the user. Thus far there is no general method available for specifying the system's representations. Structuring representation has to start by

analysing what has to be represented. In situations of groupware, this may include the system state and dynamics, relevant elements of transaction history, and metacommunication (to assist users to maintain interaction).

DICTIONARY OF BASIC TASKS

ENTRY 1:
[TASK > CANCEL MEETING]
[EVENT > CANCEL MEETING]
[OBJECT > CANCEL_MESSAGE]
[OBJECT > TODO_LIST]
T1 [EVENT > CANCEL MEETING]
[OBJECT > TIMESLOT: *t]
"cancel meeting in current timeslot"

PRODUCTION RULES

Specification Level

T1 [EVENT > CANCEL MEETING]
[OBJECT > TIMESLOT: *t]
::= **specify** [EVENT+**specify**[OBJECT]+**end-input**

Reference Level

specify [EVENT > CANCEL MEETING]
::= **select'button** [CANCEL MEETING]

specify [OBJECT > TIMESLOT]
::= **name** [TIMESLOT: *t]

Lexical Level

button [CANCEL MEETING]
::= [%COMMAND_BUTTON%: "cancel meeting"]

name[TIMESLOT: *t]::= [%STRING%: *t]

Keystroke Level

select [%COMMAND_BUTTON%]
::=M**OUSE POSITION**[%COMMAND_BUTTON%]+
CLICK-LEFT-BUTTON

name[%STRING%: *t] ::= **KEYS**[%STRING%: *t]

Figure 1.b: Example ETAG specification (continued)

A representation of the current state of the system should be available to provide the user with all knowledge of the UVM including the relevant complexity of the network and information about the presence and state of actors.

The dynamics of the system to be available for representation may include, as far as relevant, information about ongoing processes of task delegations, coordination activities, and communication traffic.

Representation of the history of transactions, for the groupware situation, will sometimes need to concern not only interaction between the system and the current user, but also transactions in which other users are involved, temporal aspects of network traffic and dynamics of complex systems (e.g. when multiple intelligent agents are involved in performing complex and interrelated tasks).

Metacommunication consists of a special type of interaction that is maintained for the sake of enabling smooth human-machine collaboration. Either the system or any user could need to communicate about aspects of task delegation and coordination. Metacommunication may concern either the task domain, system functionality, interaction language, or representations.

c. *The designers' representations during design.* We experiment with an OO notation. We endorse the advantages generally subscribed to OO-methodologies (van Vliet, 1993). Two of them have a direct bearing on designers' representations:

- The OO approach is more natural. It fits the way we view the world around us. The concepts that show up in the design have a direct counterpart in the UoD being modeled, thus providing a direct link between the design and the world being modeled. This makes it easier for users to comprehend the design and discuss it with the analyst.

- The OO approach focusses on structuring the problem rather than any particular solution to it. The result of an OO (task) analysis and design is a hierarchy of objects with their associated attributes which still resembles the structure of the problem space.

Task:	Cancel Meeting
relevant objects:	meeting m participation list l cancellation message c timeslot t todo_list d
actors / roles:	meeting participant
task relations:	one of: - forward meeting cancellation (role: secretary) - receive meeting cancellation (role: member of participation list)
initial state: timeslot t blocked with meeting m	**final state**: timeslot t empty
precondition: meeting initiator agrees on cancellation	**postcondition:** delivery of cancellation message is confirmed
task structure: basic task	
user actions: - initiate cancellation - indicate timeslot t	**system events:** - send cancellation mess. c to all participants on participation list l - copy message m to todo_list d - empty timeslot t
comments: Meeting initiator cancels meeting	

Figure 2.a: Example OO notation of task

In our design exercises we apply an OO version of ETAG as well as an OO notation for task models - up to now we do not have an OO interface to prototyping tools (we use Visual Basic) but we definitely feel the need to have one.

Figure 2 illustrates our current notation (van der Veer, Lenting, Bergevoet, submitted). Tasks are described in an object oriented way. After the task name, the first part indicates semantic relations (to objects, actors and roles, and other tasks). The next part specifies the task, by providing relevant descriptions of the initial and final system state, relevant conditions for initiating the task and considering it completed, and decomposing the task into subtasks, user actions, and system events.

Object: timeslot	
superordinate: personal text slot	**subordinate:** day_slot hour_slot quarter_hour_slot
themes: one of - meeting - holiday - business travel, empty	
places: month calendar week calendar day calendar	
relevant tasks: cancel meeting initiate meeting postpone meeting receive meeting cancellation forward meeting cancellation	
actors/roles; competence meeting participant; initiate cancellation meeting initiator; prohibit cancellation	
passive/active: passive	
attributes: time, date	

Figure 2.b: Example OO notation of task related object

The task related object illustrated in Figure 2.b shows, after the object name, a part that is intended to specify the semantics of the object in relation to other objects, and a part indicating the semantic relations between the object and tasks and actors. The example is again taken from a class exercise on designing a calendar and meeting organiser.

2. THE ART OF DESIGN FOR PEOPLE

The people for whom systems are developed can be distinguished in two classes, (1) the clients, i.e., the people or organisations that pay for the design or acquisition of systems, and (2) users, i.e., the people or groups that apply the systems as part of their daily work, often refered to as the "end-users". Throughout the process of design, these two classes of "users" have to be distinguished, since they may well have different goals regarding the system, different (and possibly even contradictory) knowledge about the task domain, and different views on what is an optimal or acceptable system. This, however, does not mean that in certain situations these classes will not overlap. But even if this is the case, individual persons may well turn out to have contradicting views on the system they need.

Designers are, in relation to these two classes, in a situation of potential political stress. The two classes have contradicting input in the specifications of the system. Moreover, the financial and temporal constraints on the amount of effort to be invested in designing the different aspects of the system (like specifying functionality and user interface, implementation and testing) will tend to counteract the designers' needs to sufficiently take care of the users' needs.

These problems illustrate the meaning of Hirschheim and Klein's order-conflict dimension. The design approach illustrated in this paper does not provide a solution to political problems of this kind, but it makes the different aspects and their relations explicit, enabling the designers to be aware of the problem and of the alternatives to be considered for solution.

Making a distinction between the two classes does not attack the problem of diversity of users. In the groupware systems we consider, we have to face the existence of end-users playing different roles, as well as of end-user groups that, as a group, have knowledge or a view on the task domain that may not be equivalent to the (average or aggregated) knowledge and views of the individuals.

2.1 Design as a temporal activity

There exists a vast number of design methods. They generally consist of a set of guidelines, heuristics, and procedures on how to go about designing a system. Next to that, they offer a notation to express the result of the design process. Different design methods place a different emphasis on the process and notational aspects. Together, the process and notation provide a *systematic* means for organizing and structuring the process and its products.

Design methods generally suggest a rather strict ordering of activities: first do this, then do that, etc. This holds for top-down methods, bottom-up methods, and middle-out methods (like OO) alike. However, design is a problem-solving activity, and as such very much a matter of trial and error. In the presentation of a mathematical proof, subsequent steps dovetail well into each other and everything drops into place at the end. The actual discovery of the proof was probably quite different. The same holds for the design of software. We should not confuse the outcome of the design process with the process itself. The outcome of the design process is an (a posteriori) *rational reconstruction* of that process.

Design exhibits a "yo-yo" character. Something is being devised, tried, rejected again, new ideas crop up, etc. Designers frequently go about in rather opportunistic ways. They frequently switch from high-level application domain issues to coding and detailed design matters, and use a variety of means to gather insight into the problem being solved (Guindon and Curtis, 1988; Rosson et all, 1988). For example, if a designer recognizes a partial solution to some part of the problem, he will immediately enter a detailed design stage to work out that solution. The other way round, the development of a solution may lead to the discovery of new or additional requirements, which then become the immediate focus of attention.

As such, a temporally well-ordered sequence of design activities seems to be a special case for well-structured problems when the designer already knows the correct solution. Deviations from this well-ordered sequence then are a natural consequence of the ill-structuredness of many a design problem during the early stages of its design.

2.2 Design as an activity structure

Viewing design as a structure of interrelated activities, we need a framework that guides management of the activities. We start from the well known fact that development of complex systems is an enterprise, where several groups of people are involved.

Since the system to be developed will feature in a task situation, we need to model this, including the 3 viewpoints mentioned in 1.2, i.e., the people (users and user groups), the tasks and task structure, and the situation. In our approach we further structure this activity into the development of two different categories of models, which we label task model 1 and task model 2. The first one models the "current" task situation, i.e., the actual structure and organisation of activities including the artifacts and technologies that are available and used. In order to develop task model 1, several methods of knowledge acquisition have to be applied, as refered to in section 1.1.

Task model 2 models the task domain of the future situation, where the system to be developed would be used, including

changes in organisation of people and work procedures. The relation between task model 1 and 2 reflects the change in the structure and organisation of the task world as caused by the implementation of the system to be developed. As such, the difference is relevant both for the client of the development process and for the user. The development of task model 2 from task model 1 is an instance of specification, in this case based on knowledge of current inadequacies and problems concerning the existing task situation, needs for change as articulated by the clients, and insight in current technological development.

The information system to be designed, is specified as a concequence of task model 2. It has to be modeled in all details that are relevant to the users (hence the "user's virtual machine") including cooperation technology and user relevant system structure and network characteristics. In this perspective we need to specify the functionality (as modeled, e.g., by the semantic level of Moran (1981) or by the object structure and basic task structure of ETAG (Tauber, 1990). The other activities in relation to the specification of the UVM concern the interaction between user and system, with its two directions of communication that we will label the "language interface" (modeling the language in which the user expresses himself to the system, see 1.3.a) and the "presentation interface" (modeling the system's actions and representation of relevant information for the user, see 1.3.b). These 3 components of the UVM have to be distincted in the activities of design. They lead to 3 models that are strongly related. The language interface and the presentation interface are the two sides of the coin regarding user-system interaction, and both express, each in its own direction, the semantics and functionality of the system. Hence, during design the 3 modeling activities have to be coordinated and continuously controled for consistency.

The specification of the "new" system (the UVM) will have to be fed back to task model 2. If design decisions made the specifications to deviate from the original specification, this has to be considered explicitly. Changes in the model of the system may result in changes in other parts of the task world, like organisation of people's work, communication structure, and procedures. If this would cause task model 2 to deviate from what is considered an adequate answer to the request for system development, this could lead to reconsidering the whole set of decisions taken sofar. Feedback is another activity that has to be coordinated and continuously controled.

The specification of the new system includes many design decisions that have to be considered in relation to the prospective use. The activity of evaluaton will therefore be mandatory in parallel to each of these activities. In some cases of design decisions, guidelines might be of help, in other situations formal evaluation may be applied. Formal modeling tools like CCT, TAG, or ETAG may provide an indication of complexity of use or learning effort required (de Haan et al., 1991).

In many cases of design decisions, however, evaluation can best be done by confronting the relevant aspects of the intended system with the understanding, behavior, and feelings of the future user. Some kind of prototyping will often be the best way to confront the user with what solution is proposed. A prototype could allow experimentation with selected elements of all models that are developed for aspects of the UVM. It may enable imitation of (aspects of) the presentation interface, enable the user to express himself in (fragments of) the interaction language, and can be used to simulate some aspects of the functionality, including organisational and structural characteristics of the intended task structure. Hence, prototyping and related early evaluation is another activity that has to be explicitly distinguished.

2.3 A team model

For the past 4 years we have been performing design excercises that were intended to cover the whole set of activities mentioned in the previous section. Based on this experience, during courses in user interface design at a university and a technical university and in post academic courses for industrial system designers, we developed a view on structuring our design team in relation to the design activities mentioned.

Figure 3 illustrates our team model. It seems worthwile to make distinctions between major activity clusters and assign each to a separate (sub) team:

- Task model 1 is developed on the basis of knowledge acquisition activities.
- Task model 2 is the result of analysing task model 1 on its values and problems for the user. This is a specification activity taking account of the clients' request and relevant knowledge of state of the art technology.
- The specification of the UVM can further be subdivided into
  - modeling the functionality;
  - modeling the presentation interface;
  - modeling the language interface;

  where the development of the UVM is based on another specification activity on the basis of task model 2, and where design decisions have to be fed back to task model 2.
- Prototyping is based on specifications that are derived from specific or general evaluation needs that emerge during design decisions on the UVM. Prototyping will return evaluation to the previously mentioned modeling activities.
- Implementation could be the task of another team, based on the specifications that eventualy result from the UVM when evaluation has been satisfactory. In fact, we never actualy performed this activity during our exercises.
- As already mentioned, in the case of distinguishing these activities a lot of "traffic" will have to go between the different teams that take responsibility for certain parts of the work. Consequently, another activity that could be identified, and that in practice turned out to be of major importance, is the management and coordination. Management includes controling feedback, guarding consistency between the different aspects of the UVM, monitoring evaluation, and controling specification as it takes place in various transfers between activity clusters.

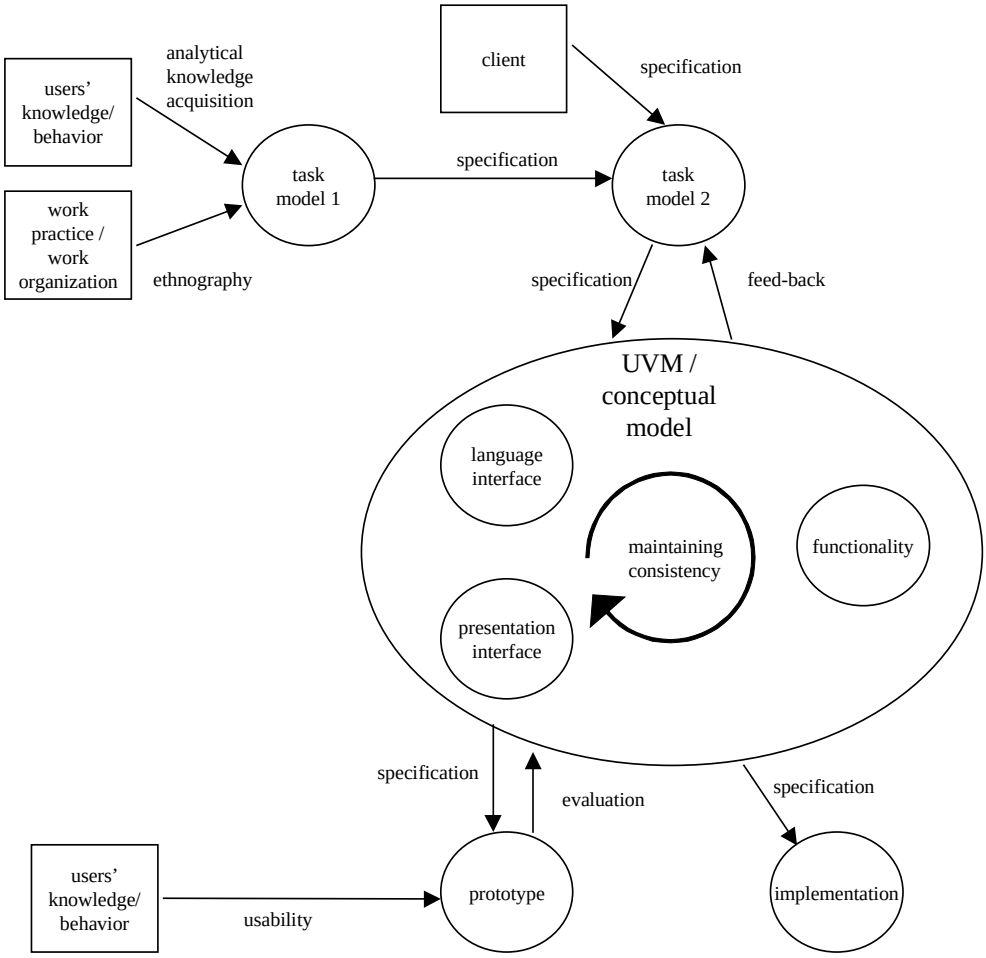

Figure 3. Design team, structure of activities

3. PRACTISE OF TEAM DESIGN

In this section our experience will be illustrated, showing both the need for multi-disciplinary activities in groupware development, and the advantage of an interdisciplinary team. We will show the different methods currently available for modeling the different aspects of design, and the requirements needed to integrate them. We will provide practical examples of modeling and prototyping and methods to coordinate and evaluate.

3.1 The basic problem: multi-disciplinarity

The main problem of the design activities as indicated in the previous section concerns the fact that different approaches seem to provide conflicting viewpoints on what methods can be used and what goals can be accomplished. Classical HCI tends to lead to Taylorism, neglecting the reality of multitudes of goals and multitudes of methods in any task domain. On the other hand, sociological and ethnological approaches towards groupware design tend to omit analytical methods and formalization, preventing detailed records of design decisions and decision structures. Still, both extremes in this conflict provide unique contributions.

In order to design for people, we have to focus on two sides of the "coin": both on the individual users of the system as well as on the client, and on the structure and organisation of the group for which the system is intended. Whe need to know the individuals' knowledge and views on the task and on applying the technology and the relation of using technology and user characteristics and task-relevant user characteristics (expertise, knowledge, skills). Regarding the group, we need to know the structure and dynamics, the phenomena of group "knowledge" and work practice and culture. These aspects are needed in order to acquire insight in both existing task situations and in projected task situations where (new) cooperation technology is introduced. Both types of insight are also needed in relation to design decisions related to the UVM, both regarding functionality and the user interface. Consequently, in prototyping and field testing we need insight in both acceptance and use of individuals, and in effect of the new design on group processes and complex task dynamics. Figure 3. Design team, structure of activities

3.2 Combining forces

In our practice of teaching design we experimented with an eclectic approach, in an attempt to develop structures to combine valuable methods (van der Veer, Lenting, and Bergevoet, submitted). The general framework for our approach is the design activity structure depicted in figure 3. The activity of building task

model 1 is based on both knowledge of the single user (psychological variables, task related variables, knowledge and skills), and on complex phenomena in the task situation (roles, official and actual procedures, variation in strategies and situatedness of procedures). The integration of this insight in a formal model will often not provide a single (or a single "best") decomposition of tasks and a unique structure of relations between people, activities, and environments. The model will often show alternative ways to perform a certain task, role-specific and situation specific methods and procedures, and a variety of alternative assignments of sub-tasks to people.

From this, and from the specifications that arrive from the client's requirement, compromises will often have to be found in defining the new task situation for which the technology has to be designed. This process will include interpretation of problems in the current task situation, negotiation with the client regarding his conditions and the resources available for design (including both financial impacts and time constraints). Ultimately, decisions will have to be made regarding complex aspects as re-arranging power influences and control possibilities of future users in various roles. As far as the client allows, flexibility and alternative procedures can be provided, and certainly situatedness of procedures will have to be explicitly designed. The semi-formal languages we experimented with (in an OO notation) can be formatted in such a way as to enforce decisions on these aspects.

Again, when detailed design decisions are being considered, early evaluation will need to include both analytical methods (formal evaluation and cognitive walk-through techniques) in combination with situated usability testing where users of different roles need to be studied both in the sense of traditional individual measures and in the sense of ethnographic interaction analysis.

3.3 Interdisciplinarity can be learned

In our exercises on design we followed the activity structure from figure 3. Design teams were formed that included expertise from different disciplines, in all cases including computer scientists and software engineers, often working in combination with psychologists, classical ergonomists, specialists in artificial intelligence, and / or electrical engineering. None of our design teams consisted of less than 3 of the above listed disciplines, but the actual composition was given by the attendence to the design classes. We composed design teams of 10 - 30 members, and split these teams in small groups of 3 - 5 members (varying from 3 to 7 subteams). The circles in figure 3 indicate the major grouped activities that we assigned to subteams, combining "adjacent" activity centers if we lacked enough teams, e.g. task models 1 and 2, language interface and presentation interface. We did not include implementation (since this was outside the scope of the classes), and we introduced one extra subteam, responsible for management and communication between the sub-teams, and for negotiation with the client and documentation and reporting.

One activity that we mostly split into two separate sub-group tasks was the knowledge acquisition that aimed on the (individual) users, and the analysis of work practice and work organization. Hence, for task model one there were two separate streams of input, one from a psychological oriented approach and one from an ethnographic point of view, that had to be combined to define the task model. This forced people to focus on single methods in depth and still integrate their results with those of a very different approach.

For the formation of the sub-groups we always made sure there was interdisciplinarity in each group, preventing, e.g., all psychologists to perform their own expertise and all software engineers to focus on formal specifications. The background of this measure is that each designer is faced with activities that stem from "foreign" disciplines, and has to cooperate as well as to communicate with colleague designers from different backgrounds, provoking insight in the complex and interdisciplinary nature of the total activity structure.

Design exercises of the size we performed, taking up to 140 working hours per participant, showed that the first 20 hours seem to be used, among other activities, for learning to think, communicate, and act in an interdisciplinary setting, accepting new concepts and approaches from each other and becoming prepared to collaborate in this structure. From then on, experts seem to be team members of the same status as novices in a certain approach, as far as decision power is concerned, although expertise continues to act as a source of alternative decisions and alternative methods.

4. AN EXAMPLE: THE BOOK SELLING CASE

A recent design exercises in one of our courses featured the redesign of a campus book selling system. The original system offered members of student unions a service to order prescribed books at the start of each trimester. Students who ordered books this way receive a 15% discount from the campus bookshop. The university concerned has thirteen faculties each having their own student union, and one campus bookshop. The Book Commissionar of one of the faculties agreed to act as the "client" of the system re-design.

About 20 students from different disciplines like Mechanical Engineering, Computer Science, Educational Psychology, and Electrical Engineering attended the design class. Small subteams, each consisting of three students, were formed for the main activities indicated in figure 3. Care was taken to have at least two different disciplines in each team. The resulting (seven) groups took the roles of analytical task modelers, ethnographers, functional and formal designers, interaction designers (both language and presentation), prototype builders, evaluators, and management. Each group started to choose their own strategy to complete the team activities. At the start of the project management prescribed generic procedures for timelines and meetings, defined documentation of specifications and decisions, provided formats for group presentations, and arranged email traffic.

4.1 Modeling the current task

As a first team activity, involving both the analytical and ethnographic task modeling groups, task model 1 had to be established. In collaboration with management, two representative student unions were selected, viz. Computer Science and Management Science. It was decided to build task model 1 on the basis of the actual book selling system used in these two faculties. The two task groups had to cooperate to generate the taskmodel, but they decided to apply their different methods for collecting the relevant knowledge on the task domain. Several roles were identified: Student, Teacher, Book Commissionar (representing the student union), Bookshop Manager, and Student Administration . The analytical method consisted of a structured interview technique focusing on a hierarchical task structure for each role, by means of a question-answering protocol (Sebillotte, 1988) and of observation of workers who were asked to perform certain tasks for the sake of recording non-verbalizable tasks and actions. Ethnographers, on the other hand, tried to become involved in real life activities of users of the system as participant

observer, and performed "interaction analysis" (Jordan, 1994) by means of systematic interpretation of video records of relevant interactions, focusing on the complex relations between objects, people, and situations. Different from analytical approaches towards task analysis, ethnography does not prescribe any formal representation.

Both groups transcribed their own data and builded their own representations of the task structure. Till this stage cooperation was restricted, the strenght of both methods was used to its full extent.

Before an integrated task model 1 could be created it was necessary to understand each others knowledge of the task world. Therefore an activity called cross-referencing was executed where the analytical group interpreted the ethnographic transcription and vice versa. Both groups discovered that their own task knowledge was both incomplete and different. The ethnographers, for instance, had defined a set of objects (all kinds of book order forms, featuring in different situations), while the other group could only refer to these objects as "the form".

The subsequent integration activity showed some remarkable aspects with respect to the task structure that had to be modeled. The analytical group used a tree representation (one for each a role) derived by top-down analysis, while ethnography define a bottom-up analysis. Furthermore it was found that the depth (number of hierarchical levels) of analysis in the analytical trees (4) was smaller than the ethnographically derived structure (6). The latter analysis showed more detail: the actions as elements of a simple task that apparently are not verbalizable. The perspective chosen by each group also differed: separate roles (analytical) vs. work practice. Despite all the differences, the merging of collected knowledge, monitored by the management group, happened to be quite easy: The ethnographers provided both the details that the analytic group could not detect by their methods, and contributed the relevant situational aspects of task structures.

The combined task structure, of which figure 4 shows a fragment, defines task model 1. The total task tree had about 150 task nodes, many of which might be candidate for redesign. Hence, the detailed specification of the task and object templates (see 1.3.c) was not yet finished at this moment.

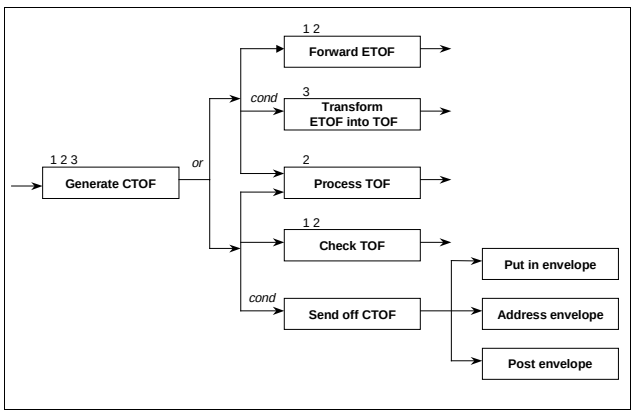

Figure 4: Fragment of task model 1, book selling case

Figure 4 shows how a Combined Teacher Order Form (CTOF) is generated in the existing situation. Main part of this is the transformation (by teachers) of an emtpy "TOF", which is again decomposed into subtasks (indicated by arrows). The task structure is clarified with the help of "constructors" like OR and COND(itional). The default constructor is SEQ(uential).

4.2 Specifying a new task structure

In order to specify task model 2, user problems were collected during the analysis of model 1. Because of the complex task world, the designers tried to generalise the problems, resulting in five major problem types. How these problems do harm the total book selling system could be detected from the tree representation of task model 1. All task nodes that somehow relate to one or more problem types are said to be infected. In Figure 4, e.g., three problem types occur: metacommunication (1), communication with teachers (2) and invalid forms (3). Problem types 1 and 2 account for about 80 % of all problems, so they should be solved first. In discussion with the whole design team several solutions were identified, by means of a design rationale method (QOC, MacLean et al., 1991). The chosen solution to problem 1 and 2, for example, was the use of email. The task tree was altered to cure the infections, resulting in a first attempt to task model 2.

This first design proposal (based on the solution of user problems) was presented to the client by members of the two task modeling groups and the management. According to (negotiated) client specifications a second proposal was generated, and presented to the technical design groups (functional and formal designers, interaction designers, together responsible for the design of the UVM). With their comments on technical feasibility a third proposal was generated, which represents task model 2. The specification of the part shown in figure 4, for instance, resulted in the fragment of the task structure shown in figure 5 (we did not fill in the node labels).

The new task structure for this part of the system shows many more nodes, which means that the task to generate CTOF has become richer i.e. has more subtasks at a higher level. As a solution to problems 1 (meta-communication) and 2 (teacher communication) the usage of email defines a series of sequential basic tasks to replace the problematic subtasks (first four boxes of the middle column). On the other hand, it is typical for the second task model that complex tasks like "Send CTOF" need only one action in the new system (single box). As a consequence of curing the problem that follows after the CTOF has become valid, some new tasks (last boxes in the middle column) have become part of the new CTOF-tasktree.

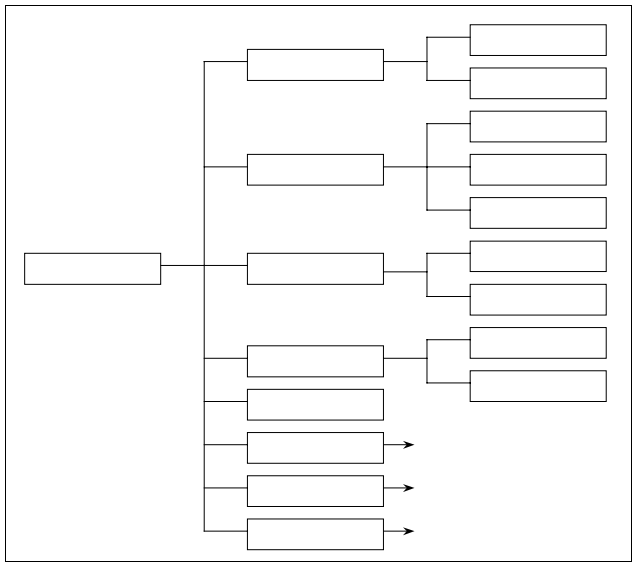

Figure 5: Task model 2, fragment equivalent to figure 4

Task model 2 was further specified with the help of an OO

notation similar to the example shown in figure 2.

4.3 Specifying the users' virtual machine

The UVM specification was in this exercise performed by two sub-teams, of which one focused on design of the functionality and formal specification of the design in ETAG. The other group focused on specifying the interaction and the representations. The two groups started in collaboration, with the reformulation of the basic (system) tasks, grouped per role, from task model 2. For each basic task an ETAG description was developed (see figure 1 for an impression of the formalism involved). The ETAG specification defines system events. This specification has to be combined with the specification of the interaction style, resulting in both an ETAG set of production rules (see figure 1 for an example) and a state transition diagram (see figure 6 for part of the interaction involved in ordering a book).

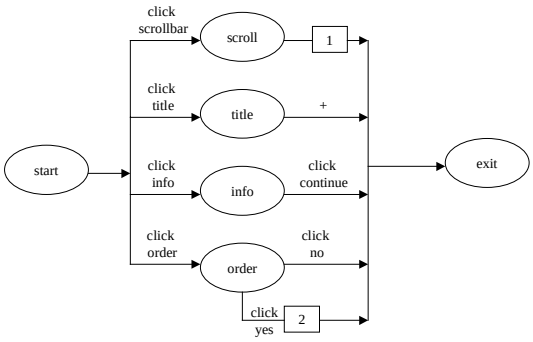

Figure 6: Representation of interaction, book selling case

The choice of interaction styles was in fact decided per role, with the help of a selection method proposed by Mayhew (1991). The proposed combination of dialogue styles for each role was discussed with the complete design team and illustrated with some screen layouts representing the look-and-feel.

In this design case the specification of production rules in ETAG turned out to be highly influenced by the proposed screen layouts. These screens (about 20 in total) evolved (using no systematic method) from the choice of style(s), not from the technological solutions (the basic events as specified in ETAG). Figure 7 illustrates one of the initial screen designs.

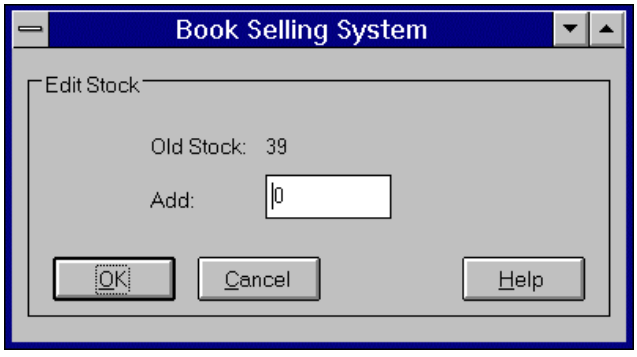

Figure 7: Example screen design, first trial

The management had to watch consistency carefuly, focusing mainly on the users point of view, instead of on expected implementation simplicity.

After a first "complete" UVM was specified, it turned out that several changes occurred with respect to the specified taskmodel 2, which needed feed back to the task modeling sub-teams. Here management was needed again to decide whether one major update of the second task model would suffice.

4.4 Prototyping and evaluation

As soon as the first specification of the interface (style and look) for each role was finished, the prototype group started to build a prototype, in this case using Visual Basic. Each first version of a prototype was given to the evaluation team for usability testing. In relative short try-outs it could be learned that some interfaces had to be changed drastically, resulting in feed back to the UVM-design teams and even indirect to task model 2. Parallel to the prototyping activities for the more finished parts of the specifications, the prototype group also wrote some user manuals, which were subsequently tested by the evaluation group in combination with the prototypes.

The evaluation group in this exercise in fact could be regarded as a user supporting group. Everything any design subteam needed to be evaluated was tested by them and, moreover, they also tested without any direct request, whenever they felt to have indications from contact with users that there might be problems. E.g., as soon as the second task model had become "final" they evaluated it by presenting the chosen problem solutions in a scenario to users representing the different roles in the book selling system. The attitudes of the different users that were confronted with the scenarios were presented to the complete design team and discussed with project management.

By means of cognitive walktroughs they also evaluated the different presentation interfaces (screen layouts) and resulting in useful feedback to the UVM-designers, monitored by management. Testing the software prototypes for usability involved several standard evaluation methods. Applying the SUMI (Porteous et al., 1993) for subjective usability of the final prototype version of the design resulted in scores on the various indices that were well above average for commercial software.

4.5 Managing design

The design process was lead by a project management group which took the responibility to guide and coordinate the different activities. Most of these activities (except task analysis) had an iterative feature. The design was refined in each iteration round, which had to be closely watched and documented. One of the major specific tasks of the management was the development of a contract with the client. This contract specified the project time schedule, budget, man-hours, etc. An important part of the contract specified the client's requirements with respect to usabilty, refering to indications of effectiveness, learnability, flexibility and attitude regarding the new system. At the completion of the exercise, the management presented the project results and produced the design documentation (400 pages, including design decisions and rationales, screen dumps, and formal specifications, as well as evaluation and test results).

CONCLUSIONS

Designing complex systems is a complex activity. Structuring the design into a set of interrelated design activities enables management of the whole. Interdisciplinarity is a prerequisit in this type of design. Different design task have to be identified, each with its own goals and methods. This results in the need to manage feed back and iteration between different stages in the specification of the system. Exercises in educational settings have

illustrated the feasibility of this approach. Future research should focus at refining and evaluating the methods and formalisms used, as wel as the management aspects of design.

ACKNOWLEDGEMENT

Students in Twente and Amsterdam and in post academial courses, who featured in the actual design teams.

REFERENCES

Guindon R. and Curtis B. (1988) Control of cognitive processes during design. Proceedings CHI '88, ACM, New York.

de Haan G., van der Veer G.C., and van Vliet J.C. (1991) Formal modelling techniques in human-computer interaction. Acta Psychologica 78, 27-67.

Hirschheim R. and Klein H.K. (1989) Four paradigms of information systems development. Communications of the ACM 32 (10) 1199-1216.

Johnson P. (1989) Supporting system design by analyzing current task knowledge. In: D. Diaper (ed), task analysis for human-computer interaction. Ellis Horwood, Chichester.

Jordan B. (1994) Ethnographic Workplace Studies and CSCW. In: D. Shapiro, M.J. Tauber, and R. Traunmueller (eds), The Design of Computer-Supported Cooperative Work and Groupware Systems. Elsevier, Amsterdam.

MacLean A., Young R., Bellotti V., and Moran T. (1991) Questions, options, and criteria: elements of design space analysis. In: Human Computer Interaction 6 (3&4), 201-250.

Mayhew D.J. (1991) Principles and guidelines in software user interface design. Prentice Hall, Englewood Cliffs, NJ.

Moran T.P. (1981) The command language grammar: a representation for the user interface of interactive computer systems. IJMMS 15, 3-50.

Pfaff G. and ten Hagen P.J.W. (1985) Seeheim workshop on user interface management systems, Springer, Berlin.

Porteous M., Kirakowski J., and Corbett M. (1993) SUMI User handbook. Human Factors Research Group, University College Cork, Ireland.

Rosson M.B., Maass S., and Kellogg W.A. (1988) The designer as user: building requirements for design tools from design practice. Communications of the ACM 31 (11) 1288-1298.

Scapin D. and Pierret-Goldbreich C. (1989). Towards a method for Task Descriotion: MAD. In: L. Berlinguet and D. Berthelette (eds), Work with display units 89. Elsevier, Amsterdam.

Sebillotte S. (1988) Hierarchical planning as a method for task-analysis: the example of office task analysis. Behaviour and Information Technology 7 (3), 275-293.

Tauber M.J. (1988) On mental models and the user interface. In: G.C. van der Veer, T.R.G. Green, J.-M. Hoc, and D. Murray (eds), Working with computers: theory versus outcome. Academic Press, London.

Tauber M.J. (1990) ETAG: Extended Task Action Grammar - a language for the description of the user's task language. In: D. Diaper, D. Gilmore, G. Cockton, and B. Shackel (eds), Proceedings INTERACT "90. Elsevier, Amsterdam.

van der Veer G.C., Tauber M.J., Waern Y., and van Muylwijk B. (1985) On the interaction between system and user characteristics. Behaviour and Information Technology 4, 284-308.

van der Veer G.C. (1994) Groupware Task Analysis: Modeling Complex Reality. In: R. Oppermann, S. Bagnara, D. Benyon (eds) Human-computer interaction: from individuals to groups in work, leisure, and everyday life. GMD-Studien Nr. 233, GMD, Bonn, Germany

van der Veer G.C., Lenting B.F., and Bergevoet B.A.J. (submitted) Groupware task analysis: modeling complexity. Submitted to Acta Psychologica, jan. 1995.

van Vliet H. (1993) Software engineering: priciples and practice. Wiley, Chichester.