Task Analysis for Interactive System and Service Design

Supplement to Master's thesis of
Renardus M.M. Scheffer
Groningen, 29 May 1995

Eindhoven University of Technology
Department of Phylosophy and Social Sciences
Eindhoven

KPN Research
Department of Service Development & Support
Groningen

Graduation commitee

Drs. R. Gobits
Department of Phylosophy and Social Sciences
Eindhoven University of Technology

Dr. G.C. van der Veer
Department of Computer Science
Vrije Universiteit Amsterdam

Drs. K. Engelmoer
Department of Service Development & Support
KPN Research

Preface

This report describes a literature study which was part of a graduation project for a degree in communication engineering at the Technical University of Eindhoven. The study was conducted at the department Service Development & Support of KPN research. Its aim was to get insight into the use of the various task analysis methods and techniques for the design of interactive telecommunication services and systems. Succeeding research, will be focused on one specific method. A computer application will be designed which can support this particular method. This literature study is a preliminary exploration of the topic "task analysis for interactive system design".

René Scheffer, Groningen, 29 May 1995

Table of contents

Preface

1. Introduction 1
    1.1. People and tasks 1
    1.2. Interactive systems and tasks 2
    1.3. Task analysis and interactive system design 2
    1.4. Aim of this document 4
    1.5. Contents of this document 4
2. Analysing Tasks 5
    2.1. Task Analysis 5
    2.2. Techniques, Methods & Methodologies 6
    2.3. Performing task analysis 8
    2.4. IWS: A HCI conception of tasks 10
    2.5. IWS components in task analysis products 11
    2.6. Comparative analysis of TA products 13
3. Task Analysis in system design 15
    3.1. Software design paradigms 15
        3.1.1. The classical waterfall approach 15
        3.1.2. The prototyping approach 17
    3.2. Integrating TA into system design 18
        3.2.1. TA within the classical waterfall life cycle 19
        3.2.2. TA within the prototyping approach 20
    3.3. Software designers' need for TA 23
    3.4. Integration of Task Analysis into system design 23
4 Conclusions 25
    4.1. Summary 25
    4.2 Further research 25
Literature references 27
List of Acronyms 39

Appendix A: Data collection techniques
Appendix B: Task analysis, techniques methods & methodologies

1. Introduction

1.1. People and tasks

In every day life, at work and in private, people perform tasks to achieve goals which are imposed to them by themselves or by their environment. The goals of these tasks always imply some affection of a particular part of the world. Tasks can be seen as more or less well defined mechanisms by which people bring about changes in the status of "things" in the world around them.

Only a small part of the world is of interest to the task. The domain of things relevant for a task consists of physical things (notes, doors, machines, other people, etc.) as well as abstract objects (addresses, bills, organisations, etc.). The type and the number of relevant objects in the domain are constrained by the environment, both physical and social, in which a task has to be performed. The set of relevant objects for the task is called the work domain.

People use tools to help them to carry out a task. Tools enlarge the possibilities in performing tasks. They are designed to let people carry out tasks more efficiently and carry out tasks that were previously not possible. In order to be useful, the tool has to match with the work domain of the task. The designer of the tool must consider both the characteristics of the relevant objects for the task, and the constrains, competence and preferences of people and the environment in which a task has to be carried out. These aspects determine the usability of the tool for a specific combination of users and tasks.

1.2. Interactive systems and tasks

Computer systems, or more generally interactive systems, are special tools in that they are capable to handle or alter abstract domain objects. In order to do so, both abstract and physical objects have to be "computerised", so the computer is capable of changing the objects in the domain.

By its nature the "computerised" work domain is only a representation of the "real" task domain. It is people who give meaning to this representation of the work domain. The user has to map his/her own perception of the task domain on the computerised work domain. Therefore the software has to give cues to the user to let the user form a good conception of the computerised task domain and the activities that the computer can perform in it. A concept that fits well within the task is easily adapted and explored. The ease and accuracy with which the user can map task activities on to functionality (computer activities) is a measure of usability. (Edmonds and Meech (1994)).

The mapping of the computer work domain on the human task domain introduces some extra tasks or activities. For example, the user has to specify the appropriate computerised environment for the computer. He must specify which application to use. Also the user must specify the domain objects, e.g. select the appropriate text file, record, drawing, page, word etc. These tasks enable the computer to recognise the appropriate task domain in which to work. Good software design minimises the number of enabling tasks that are need and maximise the number of work tasks that the computer can handle.

One important factor that influence human capability of performing a task in a computerised task domain, is the conceptual representation of objects and tools in the computerised task domain. To demonstrate the importance of representation, an example will be given from every day life (see also Edmonds and Meech (1994)). Consider the task of opening a tin of paint. To remove the lid from the tin, one must use a tool with the appropriate characteristics. A screwdriver might do for such a task, it has a flat blade and a shaft long enough to provide the needed leverage. A simple functional description of the screwdriver is not enough to identify it as suitable for opening the tin. The representation of the screwdriver and the tin are crucial. It enables the task performer to identify the screwdriver as a suitable tool. The better the work domain representation matches with the expectations that the user has on basis of his conception of the task, the more likely it is that the user will identify functionality as suitable for the task

An other aspect is the dialogue. The dialogue that the user has to perform determines the degree of freedom in which he can perform tasks in the computerised task domain. The interface has to control the interaction flow in such a manner that it minimises the number of enabling tasks, and optimises the freedom in which tasks can be performed. It is the art of good interface design to find the balance between the freedom in task performance and the actual control on the situation.

The flow of interactions has to be consistent with the created concept of the computerised work domain. The used interaction style must support the expected task activities of the user at a certain point in the task. For example; menus have to be used when choices can be made between computer domain activities; forms are preferred when parallel input (independent questions) is demanded while question/answer dialogues are best when sequential input (dependent questions) is asked; multi-windows have to be offered when two tasks have to be performed at the same time, etc..

From the above it becomes clear that the functionality, the static (conceptual representation) and the dynamic (interactions flow) structure of interactive systems have to suit with the users conception of tasks. Therefore, a good picture of human task performance is very important for the design of interactive systems.

1.3. Task analysis and interactive system design

The following task related activities have to be addressed when designing interactive systems:

1) The identification of the task domain and the constraints, competence

and preferences of people and the environment in which a task has to be carried out.

2) The identification of those sub tasks/activities which can be most efficiently performed by computers.

3) The design of keen computerised task domain models (data structures) which allow the computer to perform a maximum number of these sub tasks.

4) The identification and building of processes which enable the computer to perform these tasks.

5) The building of user interfaces which bridge an unavoidable gap between a user's needs, his conception of the task, and the computerised task model. Good interfaces minimise the number of enabling tasks by matching the computerised domain with the users domain. This matching implies:
   - Creating a concept of the task domain that fits as good as possible with the users domain, presenting the domain objects in an appropriate way so it is easy for the users to understand their meaning and their characteristics and to map computer activities (functionality) on the task activities.
   - Designing dialogues that give the user the appropriate degree of freedom in the computerised task domain, offering all the necessary opportunities to perform the task without loosing the control on the situation.

The activities two, three and four are traditionally the subject of the field of System Engineering (SE). Activity five is considered as the subject of the field of Human Computer Interaction (HCI). Activity one is conducted by system engineers HCI experts, or organisation experts.

Both HCI and SE have developed their own methods for acquiring relevant task information. SE uses System Analysis (SA) methods while HCI uses Task Analysis (TA) methods. Parts of some SA methods are often confused with TA methods because both analyse and model task activities. However TA differ from the data capturing modules of SA in their goals, output and focus. The differences are shown in table 1.1.

Table 1.1: Differences between task and system analysis

	System analysis	**Task analysis**
Goal	Input to design of software processes and data structures	Input to design of user interface
Output	Functional specification and system architecture specifications	User interface specifications (in addition with style guides)
Focus	Technical information processing limitations, data characteristics, and system architecture considerations	Human information processing limitations, user characteristics, and task considerations
Main objects of analysis	Data structures (computerised task domain) and functionality (domain activities by the computer)	User tasks concepts and activities

Source: Mayhew, D.J. (1992)

The application of task analysis methods provides the analyst a "blueprint" of human involvement in a system, building a detailed picture of that system from the user's perspective of the task. Parts of the task world are modelled in a more or less formal way. Both the model it self and the process of modelling give the analyst insight into human task performance. The task model can be developed prior to, along with and after the design of interactive systems, in all cases it is used differently:

- TA can be used as input to all the design activities which are mentioned earlier in this paragraph. In these cases a model is built of an existing task. In these cases the TA is used as a *source of information*.
- TA models used along with system design is novel, and allows the designer to develop the scenarios of the interaction along with the development of the rest of the system. This is useful when prototypes are built. The model has to be adjusted for each new version of the prototype. In these cases the TA can be considered as a *means to discussed the impact of design decisions on human tasks*.
- TA can be used as a means to check how well the system matches with the user's tasks. In these cases the analysis is performed after the design of the system. It is a *means to evaluate* the effectiveness and efficiency of *the user interface* and a means to perform error analysis

This report is concerned with methods that result in a more or less formal and structured description of certain (elements of) tasks. Of course, it also possible to acquire and analyse information about a task in a less formal and structured way. For instance one can solely use general data capturing techniques (interviews, observations, literature studies etc.) or even only look at critical incidents by asking people. In some cases this may be suitable. However in most cases this will result in an incomplete and ambivalent picture of the task on which it is very hard and dubious to base design decisions.

At the moment explicit task analysis methods are hardly used at KPN. It can be argued by managers, engineers and others involved in the design of interactive systems, that the human element within a system is already implicitly included in system designs. Many System Engineering design approaches involve capturing requirements from organisational needs and sometimes even user needs. Such needs represent user tasks in some way.

While these arguments are true to some extent, it is unlikely that the human element in the systems will be optimised. Task analysis methods offer a structured way of acquiring task information. Such structured information ensures that there is compatibility between user capacities and needs for performing the task, and systems goals and input needs. Therefore TA methods should take a prominent role in system design and evaluation. Usage of explicit task analysis methods can lead to more efficient and effective integration of task elements into interactive systems and services.

1.4. Aim of this document

The aim of this literature study is to get insight into the use of structured task analysis methods and techniques for the design of user interfaces of telecommunication and information technological services. The literature study

had the following goals:

- Make an inventory of the various techniques, methods and methodologies found in the literature for the acquisition and analysis of task information.
- Analyse when and how task information can aid the design of interactive systems, especially the user interface.

This document is written for designers, managers and usability experts of KPN who want to know more about the use of task analysis for the design of user interfaces of telecommunication services and information technology. It is not the intention of the document to discuss one method or a category of methods at length. Those who want to know more about one specific method or technique, are referred to the appendix B where an extract is given for the most referred techniques and methods in the literature on HCI. There is also an reference to a more thorough discussion, for each of the methods.

1.5. Contents of this document

Chapter two intends to give an overview of the various techniques, methods and methodologies that are proposed in the field of HCI for acquiring task information. At the beginning of the first chapter some general classes of techniques, methods and methodologies will be discussed. Afterwards some important general issues on performing a Task Analysis, will be discussed. Also a framework of Whitefield and Hill will be presented which enables a comparison of TA products. In the framework components are defined that can be presented in a TA model. By analysing how the components are presented in the various methods, comparison of TA models become possible. As an example the framework will be used to compare four task analysis products.

Chapter three is dedicated to the various design issues and the role that task analysis should take in it. At first the two most used paradigms for structured software design will be discussed, to get a clear picture of which issues are handled when in the design process. The use of TA for all the phases of each of the paradigms will be discussed in general terms. Next the work of Sutcliffe (1989) will be discussed. Sutcliffe attempted to make a taxonomy to map the various task information collection techniques on the design issues in the different phases of the design process. In the next paragraph, a design strategy as proposed by van der Veer (1994) will be presented which combines the use of task models with prototyping in order to design complex interactive systems. Finally, some problems will be discussed which may occur when using task analysis information as input for a system analysis.

2. Analysing Tasks

2.1. Task Analysis

"Task Analysis" (TA) is not a term with a fixed and agreed meaning. It is a means to collect information about a task, organise it, and then use it to make various judgements or design decisions. The term task analysis covers a range of methodologies, methods and techniques used by managers, trainers, procedures writers, safety assessors, ergonomics practitioners, system designers and human factors specialists. A small sample of topics which can be supported by TA:

- Job specification
- Staffing and Job organisation
- Training design
- Writing manuals and guides
- Human reliability assessment
- Safety management
- Performance checking
- Allocation of functions (between humans within an organisation or between humans and machines)
- Man-machine interface design

In this report the possibilities and needs for TA to support IT and telecommunication services design are investigated. Therefore the topics are limited to man-machine interface design, allocation of functions between humans and machines, and performance checking. All of these topics are covered by the fields of System Engineering (SE) and Human Computer Interaction (HCI). I will focus on TA from the perspective of HCI. Most of the TA methods in this field focus on task information for the design and evaluation of the User Interfaces (see also table 1.1).

There is no real consensus among HF/SE practitioners concerning what TA is and this promotes a lot of confusion. For some people it is concerned with gathering information about tasks, for others it is about presenting that information. Therefore Whitefield (1994) proposed to make a distinction between TA products and TA processes. A TA *product* is the result of a task analysis. Most of the time this will be a model of parts of the task world. A TA *process* is the means by which the product is constructed. It is the sequence of activities that the task analyst has to perform.

2.2. Techniques, Methods & Methodologies

How can information concerning tasks be captured and presented in a structured way? A fundamental distinction has to be made between three structured manners of acquiring and handling task information:

1) **Task modelling techniques:** These techniques model one aspect of a task

for analysing a specific purpose, like interaction analysis (CLG, TAG), or performance analysis (GOMS) or learn ability (CCT) of a user interface. The models are created in an analytical manner. The technique only describes a notation technique. It demands certain skills to create and understand these models. The models created with these techniques are very detailed. Sometimes they can be used to specify a certain aspect of a new user interface, as well as to evaluate an old one.

2) **Task analysis methods:** Task analysis methods exist of at least one technique of describing certain aspects of a task (a TA product), as well as a procedure how this description can be obtained via general data techniques (a TA process). In a certain way one can compare the TA products with the task modelling techniques discussed above. However most of the time the task analysis methods are used with a less specific purpose than the task modelling techniques. They describe the task as a whole rather than focusing on solely the information technology. Most of the methods derive a hierarchical decomposition of tasks. However the methods differ in what to decompose exactly. Some methods model task activities (HTA, ATOM), others knowledge structures (TAKD, KAT, MAD). Most of them also derive some procedural (task sequences) and declarative information on the task. A few a methods even relate s object structure to the task (KAT, ATOM). Some task analysis methods have a theoretical background (KAT, CTA). Others were developed from an empirical basis (TAKD, HTA, ATOM). The Task Analysis methods can be used as a source of information in the analysis phase of the design process.

3) **Social-organisational methodologies for requirement specification:** Some writers consider these methodologies as high-level 'macro' task analysis (Benyon (1992), Sutcliffe (1989)). However these methodologies do not provide a task model, nor are they focused on tasks solely. They use a phased, sometimes iterative, approach for capturing system requirements. They derive not only user requirements but also organisational and technical requirements. Most of them do not concentrate on tasks but try to capture general, not task specific characteristics of users.

The different techniques, methods and methodologies acquire task information on a different scale. The modelling techniques analyse tasks on very specific aspects. The methodologies give only very general information on tasks like who are the task performers and what are the characteristics of the environment in which the tasks is performed. The TA methods are somewhere in between. It is possible to use the product of a TA method as input for the creation of modelling techniques. Also TA methods can be used for capturing information in a social-technical methodology.

If the term task analysis (TA) is used in this report, it refers to the output of the techniques as well as the methods. If a statement is also valid for the macro level task analysis methodologies, this will be mentioned explicitly .

A summary of the most important techniques, methods and methodologies is given in appendix A. This summary has not the intention to describe the methods at length. It only discusses some important features of each of the task analysis techniques. Also a literature reference is given, to a more thorough discussion.

In the succeeding paragraphs of this chapter, some generic components of both the TA process (2.3) as well as the TA product (2.5-2.6) will be discussed. This discussion enables those who want to use TA, to distinguish the various methods from each other and to recognise the important features of a method or technique. In paragraph 2.4. a HCI conception of tasks will be discussed. This conception will be used as a basis for the paragraphs 2.5. and 2.6.

2.3. Performing task analysis

If the TA processes of the various methods are compared, it is possible to identify five general stages:

- Preliminary analysis
- Collection of data
- Modelling the task
- Analysing the task model
- Specifying the result of the TA

Sometimes the stages are not explicitly separated, but the activities in the stages are somehow represented in almost all methods.

1) **Preliminary analysis**: Before the task analysis can start, it is important to get a good picture of the characteristics of the physical and social environment in which the task is performed. Also it is important to know the purpose of the TA. This stage falls beyond the scope of most methods. However it is crucial to the success of the TA. The following questions have to be answered:

- What is the purpose of the TA? Is the analysis meant to capture system requirement information, or to make a concept for a new interface? Will the analysis be used as input for system analysis, or is it meant for system evaluation? Must there be any judgement about the performance of a system, or about a style of interaction? The purpose of the analysis determines which task analysis methods can be used, and the kind and quantity of the data has to be collected
- Which task has to be analysed? This may seem to be a trivial question, but it is of great importance for the data collection. It determines on what to focus when an observation has to be done, which questions have to be asked in an interview, etc.. Also it is important to define the task clearly, so that it is possible to recognise the same task in different environments, performed by people with different roles. Explicit formulation of the task analysis its purpose, enables the analyst to check if the goals are met after the analysis.
- What are the characteristics of the work domain that is to be analysed? Under which circumstances is the task performed? It is important to know certain environmental influences that might alter the task performance. Furthermore it is important to know the relation with other tasks which may be of any influence on the task performance.
- What sources can be used for data collection? This is determined by the social and physical environment in which the task is performed and the task analysis method that will be used. The decision to use a specific collection technique can be influenced by the amount of resources it takes to analysis the data. Most data types need some preparation

before they can be used in the succeeding phase of the task analysis.

2) **Collection of data**: In this phase the data is collected on which the analysis can be based. All TA methods prescribe general data collection techniques. Task modelling techniques do not. In appendix A some important features of four often used data capturing techniques are discussed.

Different data collection techniques can give a different perspective on the task. Document studies and interviews with supervisors determine *what should be done*, interviews with operators determine what operators *think they have to do* and observations and verbal protocols determine *what is done.*

3) **Building a task model**: In this stage a model is built, based on the collected data. Most TA methods prescribe modelling techniques that have a strong empirical value in that they contain observable elements or elements where task performers can be conscious of. Since task performers are capable of recognising elements in the task model, it can be very useful to evaluate such a empirical task model with the task performers. However, this might not be possible for all kinds of task performers or TA products.

The modelling techniques use a more analytical approach to task modelling than most task analysis methods do. The models created with the use of these techniques are prescriptive. They represent the perception of the analyst on a certain aspect of the task. They contain abstract elements which are difficult to understand for someone who does not know the theory behind the modelling technique. Therefore the derivation of a model by means of these techniques demands more expertise from the analyst than the TA methods do.

4) **Analysing the task model**: In this stage of the TA process, the model from the previous phase is analysed on matters of interest. Some methods have a stepwise for doing this, others use a more heuristic approach to get the desired information. Some methods (KAT, TAKD) support the analysis by prescribing the derivation of a more abstract model from the earlier built "empirical" model.

5) **Specifying the result of the analysis**: In this stage the analysis results have to be translated to design decisions. Most TA methods deliver very general information on tasks. Therefore it is not always trivial to derive design decisions or system requirements/specifications from these methods. Modelling techniques like CLG and TAG and ETAG can be used to specify certain aspects of the user interface, on basis of information captured by use of task analysis methods.

2.4. IWS: A HCI conception of tasks

In the next two paragraphs of this chapter a framework will be offered that can be used to compare TA products on their competence for the design of interactive systems. The framework, which was introduced by Whitefield and Hill (1994), consist of a scheme of nine components which ought to be represented in a TA product somehow. The core of the framework is formed by the conception of Human Computer Interaction as proposed by Dowell and Long (1989). All framework components are defined in terms of this HCI conception.

Deriving generic components of the conception of a task based on an HCI model might be useful for two reasons:

- The model claims to cover the total area of HCI. By verifying the presence of components in a TA product, its strengths and weaknesses for HCI design are exposed. A scheme with strengths and weaknesses of TA might aid the selection of the right TA method for a specific design case.

- The terms used by different authors on the TA subject are bewildering. The same terms are used by different writers to denote different things. A small sample of terms used include; plans, methods, goals, operations, actions, tasks, sub tasks, projects and objects. Linking these terms to general components, can be very clarifying.

In this paragraph, the HCI conception of Dowell and Long will be discussed. Paragraph 2.5 discusses the nine components of the framework of Whitefield and Long (1994). In paragraph 2.6, as an example, four TA products will be compared by checking the presence of the nine components in each of the TA products.

Dowell and Long (1989) proposed a concept of HCI that makes it possible to define important elements of tasks with interactive systems more explicit. Their conception is shown in figure 2.1.

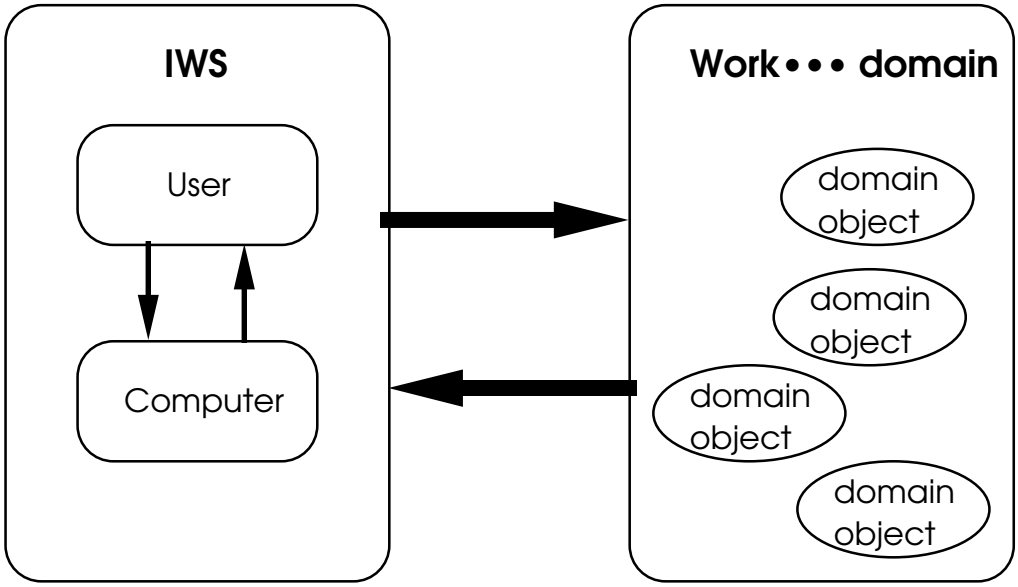

Figure 2.1: An HCI conception of task performance

In the conception of Dowell & Long a fundamental distinction is made between an Interactive Work System (IWS), a work domain, and performance.

1) **Interactive Work System (IWS):** An IWS consists of a set of behaviours, supported by a set of structures which enables those behaviours. The *behaviours* of an IWS are the behaviours that are relevant to the performance of the work for which the IWS is designed; In word processing for example, relevant behaviours include typing, reading documents, and so on. These behaviours are supported by a variety of human and computer *structures*; for example, knowledge of keyboard layout, reading skills, cut and paste facilities, document files, and so on. There is no necessary one-to-one mapping between the behaviours and the structures that support them; one structure can be used for many behaviours.

2) **Work domain**: An IWS is designed to perform within a work domain. A work domain consists of a set of physical or abstract objects relevant to the work performed. In the word processing example, objects are; letters, documents, footnotes, etc.. Objects have attributes, which can sometimes change state; for example the content or line spacing of a letter. Work can be seen as an intended state transformation carried out by an IWS within its work domain. The transformations are carried out in order to fulfil a work goal. So a work goal is a particular desired state of the attributes of the work domain. Most work goals can be decomposed into sub goals. Besides work goals there are enabling goals. Enabling goals specify desired states of IWS structures. For example; executing the correct application software, or moving the cursor to the right

location in the file.

3) **Performance**: The performance of an IWS in carrying out its work is a function of two variables;

- quality of the result (how well the final state of the domain compares with the state specified in the work goal)
- resource costs: the resources required by the IWS in accomplishing the work.

An effective IWS minimises the resource costs in performing work with a given product quality. IWS developers face the problem of designing the behaviour of the IWS such that it achieves the intended changes in the work domain at the desired level of performance. This involves specifying and implementing the user and computer structures (training, coaching, system design) that will support the intended behaviours.

2.5. IWS components in task analysis products

Whitefield and Hill (1994) used the conception of HCI proposed by Dowell & Long to determine nine components that are relevant for the concept "task". According to Whitefield & Hill these components should be represented somehow in the product of the task analysis. They stated that an ideal TA contains a description of;

- IWS behaviours
- IWS behaviour sequences,
- Task goals,
- Work domain objects and their attributes,
- Abstract IWS structures and
- Physical IWS structures.

Also a TA must contain some decomposition of goals or behaviour, a description of the performance of the IWS, and some cross-references between components. A short description of these nine component is given below.

1) **IWS behaviours**: The TA should identify the behaviours that the IWS should carry out to fulfil a work or enabling goal. The behaviours are not always associated directly with particular parts of the IWS (such as computers or users). Whether they are or not depends on the function that the task analysis fulfils in the design process.

2) **IWS behaviour sequences**: IWS behaviours occur in some order. How important the sequence is, depends on the description of the behaviour. A detailed description of physical behaviour will require a detailed sequencing. A less detailed abstract description of behaviour will not always demand a fixed subtask sequence. Tasks have to be performed but the exact order is not explicitly determined. Often the sequence is determined by the used devices in the task. However, most tasks have some sequence which is independent of the used device. For example, an e-mail message can not be sent before it has been created, no matter what e-mail package will be used.

3) **Task goals**: The TA should identify the IWS its goal in performing work. As

stated above, a task goal can be a work goal or a enabling task. To extend the e-mail example: a work goal can be "to have all received messages read" or "to have sent a reply for all messages that demand one", an enabling task can be "to be logged in on a computer system" when the e-mail program is being used or "to have an editor working" when a message is being created. A more detailed discusion of the distinction between enabling tasks and work tasks is given by Whitefield et al. (1993).

4) **A decomposition**: If the tasks of an IWS are complex enough, a decomposition into component parts is required. The TA products differ in what to decompose. Some decompose IWS behaviour, others decompose task goals.

5) **Work domain objects and their attributes**: A description of work goals requires the identification of work domain objects and their attributes which are transformed. IWS objects for electronic mail are for instance "Messages" and "Replies". These objects can have attributes like "status" (values=read/unread), "reply required" (yes/no/don't know), "date" (month, day, year), "receivers address" (RFC-822/X 400 addresses), "content" (string of characters), etc..

6) **Abstract IWS structures**: IWS behaviours are supported by various abstract IWS structures. For example; typing skills, communication protocols, text edit program's etc.

7) **Physical IWS structures**: Depending on their function, many TA products will identify the physical components of an IWS. For E-mail this can be sending user/computer, receiving user/computer. These physical components of an IWS would be included in a detailed description of the actual task of a given IWS, but may well be avoided for a device-independent description of a task for an IWS being designed.

Some TA products will also include a description of the following:

8) **Performance**: A full description of performance would also include both the quality of the desired product as well as the resource costs incurred to achieve it. The level of detail at which quality and costs could be described is constrained by the decomposition of the work goals, IWS behaviours and structures.

9) **Cross-references between components**: Good TA products should indicate how the components are related to each other. For example, it should state: which behaviours act on which work domain objects; which behaviours achieve which goals; which IWS structure support which behaviours; which behaviours occur in which sequence; which work domain objects are part of which goals; and so on.

2.6. Comparative analysis of TA products

Whitefield and Hill (1994) used the nine task components from the IWS model to compare four TA products; HTA, ICS (the product of CTA, see appendix B), GOMS and TKS (one of the products of KAT). First they checked which of the components were represented in each of the analysis. Their findings are summarised in table 2.1.

From this comparison, Whitefield & Hill derived five main features to distinguish these TA products. These features are not the only possible ones, but Whitefield & Hill considered them as most useful for the set of four TA products examined. The five features were:

1) **Psychological status of the behaviours**: How psychological real are the described behaviours intended to be, or in other words how likely is it that users do execute such behaviours in the manner described by the model? Some methods subscribe explicit cognitive behaviours (CTA/ICS, CCT). Most of the method products however have a strong empirical basis instead of a psychological.

Table 2.1: Comparing TA products on the presents of IWS components. The terminology used in each of the TA product is underlined.

	HTA	TKS	GOMS	ICS
IWS behaviours	operation - only behaviours of human described -/+ No restricted set of behaviours	actions + computer actions can be described when they are actions the user should know about -/+ No restricted set of behaviours	operators + distinguish between perceptual, Motor or cognitive acts - only user actions - set of operators is small but not exhaustively defined	description of low level information processing transactions + a restricted set of behaviours of user
IWS behaviours sequences	plan + different types of plans possible	procedures plan-> sequence of sub goals	methods, selection rules - sequence of operators described in method when more then one method is available a rule determines which method will be selected	list of transactions within and between the cognitive subsystems
task goals	goals - no difference between enabling and work goals - sometimes difficult to distinguish goals from operations	goals - No difference between enabling and work goals	goals	- not presented
a decom-position	+/- flexible decomposition + Number of levels determined by "stop rule"	+ decomposition of goals in "goal-oriented substructure"	+ Fixed number of levels: unit task level, functional level, argument level, keystroke level	- no decomposition
work domain objects & attributes	- objects not described separately - objects are easily mixed up with structures	objects + expressed in procedures & in separate "object oriented substructure" which determines the relation between objects	- not presented, can be presented implicit in (sub-) goal description	- not presented
abstract IWS structures	- not presented	- only knowledge of user, total TKS is a abstract structure	+/- Implicit user capabilities defined in operators	+ three types of cognitive subsystems
physical IWS structures	- not presented	- not presented	- not presented (except for the user)	- not presented (except for user)
Performance	+ implicit performance checking possible	- not expressed	time to complete a task	- not expressed explicit (implicit expression possible)
cross-references between components	+ operations-> plans	+ procedures-> objects + goals-> procedures	+ Operators-> goals + Methods-> goals + operators-> rules + between task levels	cognitive subsystems-> behaviours

2) **Any fixed set of behaviours**: Are there any fixed set of behaviours for use in the TA product? If they are appropriate for the task being described and for the purpose of the analysis, fixed behaviours should simplify the analysis process and support the consistency and coherence of the TA product. The problem, of course, is to ensure that the behaviours offered are appropriate in this way. The TA which have a strong psychological basis are more likely to have a fixed set of behaviours.

3) **Number of levels in the decomposition**: How many levels are there in the decomposition? It is very difficult to describe most tasks adequately at one or two levels of decomposition. On the other hand, descriptions with too many levels are unwieldy to maintain and manipulate, particularly if the different sub components of the task interact.

4) **Any fixed levels in the decomposition**: Are there any fixed levels for use in the decomposition? As behaviours tend to be decomposed more than other IWS components, this can be a similar question to whether the number of behaviours is fixed. Any fixed level is likely to reflect some decision or assumption about suitable units of analysis. A fixed number of levels makes the analysis easier to perform but it can also provide an inappropriate or misleading structure. A variable number of levels, enables the analyst to focus more on parts of interest .

5) **Separation of objects from their associated behaviours**: Does the TA product describe the objects separately from the behaviours which perform it? This is difficult, if not impossible, with many TA products; for example, HTA only describes the objects as part of the description of behaviours. In this way objects and structures which enables these objects are easily mixed up. This means that it is difficult to assure whether alternative behaviours (and thus alternative work system designs) will perform the same work.

The framework which was introduced in the previous paragraphs, enable those who want to perform a task analysis, to specify their "information demands" more explicit in terms of IWS components. The task analysis technique which delivers the most suitable information can be found by checking the presence of certain demanded IWS components in candidate methods and techniques.

3. Task Analysis in system design

3.1. Software design paradigms

To get a clear view of the contribution that TA can have to the development of systems, it is necessary to overview the various activities in the development process. In this paragraph the most used development paradigms will be discussed in order to create a framework for further discussion.

Nowadays there are roughly two software engineering Paradigms used:

- The classic life cycle or "waterfall" approaches
- The prototyping approaches

These two paradigms differ with regard to the nature of the information that a TA should provide, and when this information is needed. The two approaches will be discussed to get a clear view of the various phases in the approaches. We will use the phases as defined by Johnson (1992). In figure 3.1 both paradigms are shown.

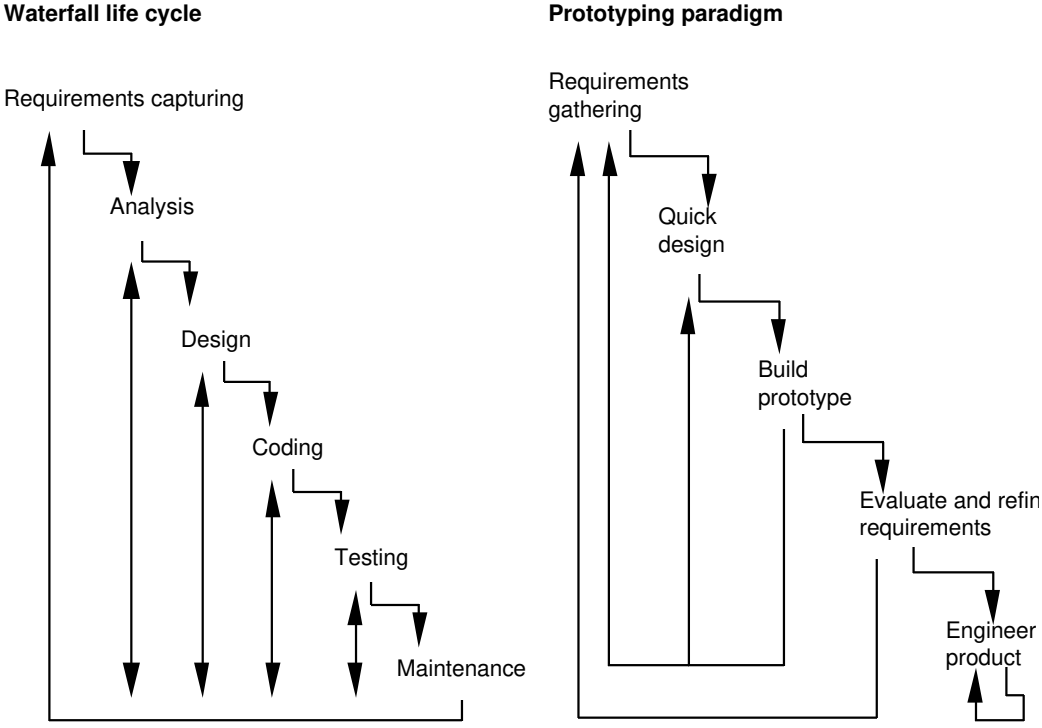

Figure 3.1: The waterfall life cycle and the prototyping paradigm
Source: Johnson (1992)

3.1.1. The classical waterfall approach

The classical waterfall approach is the oldest and most used development paradigm. It requires a systematic, sequential approach to system development. Different authors use different names for the phases of the waterfall approach. Some of them split some phases in two, depending on the focus of their discussion of the waterfall approach. (See Johnson (1992), Mayhew (1992), Sutcliffe (1989)). Most of them use, a requirements capturing

or system engineering phase, an analysis phase, a specification or (logical and physical) design phase, a coding or implementation phase, a testing phase, and a maintenance phase.

1) **Requirements capturing:** This phase involves for instance the following activities;

- An analysis of the task domain (which business activities are involved), potential users, hardware, software, all at a general level.
- Establishing user requirements (which demands do potential users have to the system)
- Establishing system requirements (which business activities, or high level user task must the system support)

The purpose of this phase of the life cycle model is to identify the scope of the system and the general design area. The output from this phase produces a general requirements document that addresses each of the areas mentioned above. Sometimes, for large projects, the first phase is called scooping and includes also some management activities like planning, or feasibility studies.

2) **Analysis:** This phase involves detailed analysis of the following:

- Tasks
- Users
- Work domain
- Software

The purpose of this phase is to understand:

- the information exchanges
- the domain entities
- the required function of the system
- the required performance characteristics
- the user interface characteristics.

Also the criteria for evaluating the resultant design are defined. The output of this phase is a specific requirements document and an analysis model of the user tasks. Also a conceptual model of the task is made that can be used as input into the design phase.

3) **Design:** This includes design of the following:

- Data structures
- Software architecture
- Procedural detail
- User interfaces

The purpose of this phase is to translate the functional specifications into a model of the software. It produces a software design specification.

4) **Coding:** The coding phase of the life cycle model involves the translation of design specification into machine-runnable form. So after this phase a runnable version of the design exists.

5) **Testing:** The testing phase of the life cycle model involves testing the design in its implemented form in terms of various requirement documents that were produced at the earlier stage of design. It includes the following:

- Logical testing of the software

- Testing of the functionality
- Testing of the usability
- Testing of the efficiency and usability of the design and the implementation

The purpose of the testing phase of the life cycle model is to assess the quality of the design and coding. It produces a report on the design quality and a set of recommendations for any redesign that is required.

6) **Maintenance:** This is the final phase of the waterfall life cycle paradigm. It is said to involve the following:

- Repairing any errors or faults in the design or the coding
- Updating the design because of changes in the requirements
- Updating of the design because of changes in the environment in which the design will be used.

The purpose of this phase is to allow the software and the design to be adapted to the changes that will occur. It produces a revised set of requirements, a revised design and revised software.

There are a number of points that can be made about the life cycle model of the system design and development:

1) Real software projects can not be as straight forward as this model assumes. Each phase may occur in a different order with iteration between phases or within a single phase. This makes it difficult for any software design/development team to follow the sequential structure of the model. It is important however, that much effort is put into the analysis.
2) It is often difficult to elicit or identify all the requirements at the start of the project because of the uncertainty that may exist in the users' and the designers' mind about the purpose of the project. This is particularly true about the requirements for the user interface. Also it is impossible to specify a perfect interface at once because the introduction of new systems in users task will change the task dramatically. It is almost impossible to make the perfect task/system match without an iterative feedback from users
3) The late production of a runnable version of the design means that any errors in the earlier phases of the design may go undetected until the system has been built.

However, there are some advantages to the life cycle waterfall paradigm:

1) It provides a comprehensive template in which many important aspects of system design can be placed.
2) The steps which the model contains are in some way generic steps that are found in most software-engineering paradigms.
3) The waterfall life cycle paradigm is the most widely used model of software design and development.
4) It is arguably better than a haphazard approach to design and development, at least from the point of view of being able to manage the production of the software.

3.1.2. The prototyping approach

When the requirements are general or not well defined, a prototyping approach is often used. An important difference between the prototyping approach and the waterfall life cycle is that the latter puts much more emphasis on analysis, making the (specific) requirements clear in advance.

Prototyping enables a designer to create an examinable model of the design. Prototypes can be runnable or non-runnable and can be simulations or evolutionary:

- **Runnable prototypes** can be either simulations of all or part of the overall design or early versions of the implementation. The difference between a simulation and an early version of the design is that a simulation cannot be evolved into the final system; it must be recoded, often on a different machine and in a different computing language.
- **Non-runnable prototypes** can be paper based or computer based. Examples of non-runnable prototyping are often found in interface design when drawing tools are used to produce sketches of what the interface might look like. One version of this is known as "story boarding", in which a number or sequence of screens are sketched and the sequence of an interaction can be illustrated by showing how each screen may change as some input is received or some process completed.

The phases of the prototyping approach are shown in figure 3.1. Prototyping involves:

1) **Requirements gathering:** In this phase the designer and the customer define the overall objectives of the system, any known requirements are identified and areas where further definition of the requirements are needed are also identified.

2) **Quick design:** Focuses on the design of the user interface, especially those aspects of the user interaction that will be visible to the user.

3) **Building the prototype:** As described above, this can be in any form. It is a more detailed version of the quick design.

4) **Evaluation**: At this stage an evaluation is carried out involving the designer, the customer and the user. The results of the evaluation are used to refine the requirements that were omitted during the initial requirements phases. Also new requirements are identified.

5) **Iteration**: Iteration occurs at the stage where the prototype is tuned, modified or rebuilt. This phase lasts until the results of the evaluations show that it satisfies the user and customers requirements and no new requirements are identified.

6) **Engineering of the product:** From the design prototype the final product is engineered. This may involve completely recoding the design in a different language and environment, or it may be a case of evolving the design from the prototype. During this phase further evaluations take place to ensure that the design is being developed in line with the requirements.

There are a number of points about the approach to design that should be

considered:

1) The customer may not be prepared to let the designer modify or throw away the prototype because they see that it is a system that could be used, even if it is not the best system as far as the requirements are concerned.
2) The designer may have used quick and perhaps inefficient solutions to build the prototype quickly and then be reluctant or simply forget to change these if the design is then evolved into the final product
3) It is essentially a trial and error approach to design.

3.2. Integrating TA into system design

The usefulness of TA techniques and methods depends on the available information at the beginning of the development process. There are three different starting situations:

- The new system will support a *new task*, also there is *no equivalent system* available. This situation may occur when new technology is introduced which need operation by human beings. In these cases it is not useful to use a TA method because most of them need an existing task to base the analysis on. However it is possible to use the task modelling techniques to "design" a task on basis of very general information, gathered in the system requirements phase.
- The new system will supported an *existing task*, however there is *no equivalent system* used within this task. This situation may, occur when parts of the task are being automated. In this situation it is possible to execute a full TA process as described in paragraph 2.3. The TA can be more useful here because the model includes user preferences, strategies and information about all kind of environmental influences. However the introduction of the system will change the task.
- The new system will support an *existing task*, also an *equivalent system* does exist. This situation may occur when a system is updated .In these cases the changes in the task model will be less far-reaching than in the case presented above. Task analysis of the old situation can reveal clues for improvement of the system.

The usefulness of TA is also determined by the paradigm that is used. When a waterfall life-cycle paradigm is used, TA models can serve as a source of information for various design issues. When the prototyping paradigm is used for system design, task models can be developed along the design process. This allows discussion about the implication of various interaction scenarios. In this paragraph, the contribution of TA to some development issues will be discussed in detail for both design paradigms.

3.2.1. TA within the classical waterfall life cycle

Sutcliffe (1989) developed a taxonomy in which various techniques for collecting task information where placed within the waterfall life-cycle. A conception of Sutcliffe's taxonomy is given in figure 3.2.

The taxonomy classifies pure task description techniques along with the user requirement modules of the system analysis methods SSADM & JDS and social-organisational methodologies for specification the requirements. The classification framework has three dimensions; a waterfall life-cycle dimension; a design issue dimension (both HCI and SE); and an analysis perspective dimension.

The *waterfall life-cycle dimension* contains the early five stages in the design life-cycle (see Johnson, 1992); a requirements definition (a early analysis to determine the scope of investigation and the appropriate user group); descriptive analysis of the current system and its users; specification of what the system should do in terms of processes and organisation; design of how the system and its users carry out tasks in logical terms; and finally physical design taking in to account environmental constraints, budget, hardware, etc.

The *design issue dimension* gives the analysis and design issues that should be addressed by the TA methods. The six design issues determined by Sutcliffe are weakly defined. This is because their definition depends partly on the life-cycle stage they act in. For instance; function/processes, changes from functional requirements to process specification and program design later in the life-cycle. The categories function/process, object/data and organisation span the whole design process from initial identification through to physical design. Other categories are more restricted in life-cycle scope. Users can not be designed, and presentation are restricted to design, using functional, data and user analysis. Striking is that non of the description techniques can be used for designing the presentation of the user interface objects.

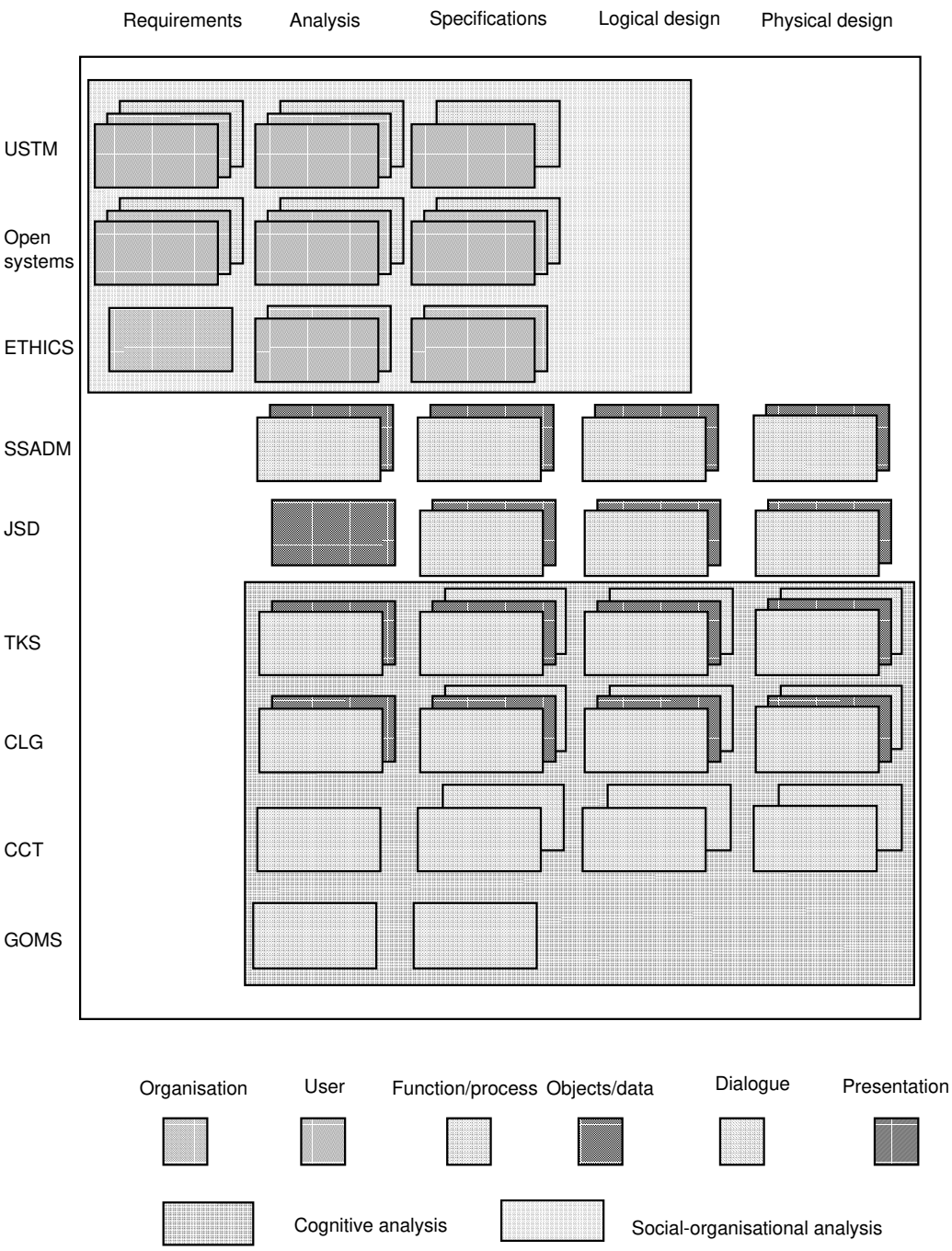

Figure 3.2: TA versus waterfall life cycle phase and design issues
Source: Sutcliffe (1989)

The *analysis perspective dimension* categorises the analysis methods by perspective of analysis. The categories are; Cognitive analysis, System analysis, Social-technical analysis. Unfortunately Sutcliffe's taxonomy does not include the task analysis methods HTA, MAD and ATOM.

The taxonomy of sutcliffe merely serves as an indication of the wide use of task analysis within the life cycle of system design. In his article he does not show how this actually done.

3.2.2. TA within the prototyping approach

Van der Veer (1994) proposes a user centred approach to system design. In

his approach van der Veer uses task analysis along with prototyping in order to specify the user interface. He defines a number of activities which can be performed by different teams of designers and analysts. Each of the activities focus on an different aspects of the design. The activities are strongly related to each other. The output of one activity serves as input for an other. There are also iterations possible between activities. A conception of the complexity of activities as described by van der Veer is given in figure 3.3.

In his approach, van der Veer uses three types of task models: a model which describes a current task situation, a model which describes the task situation after the new technology is introduced, and a model which specifies how conceptual objects and conceptual operations as represented in the information technology are related to basic human tasks.

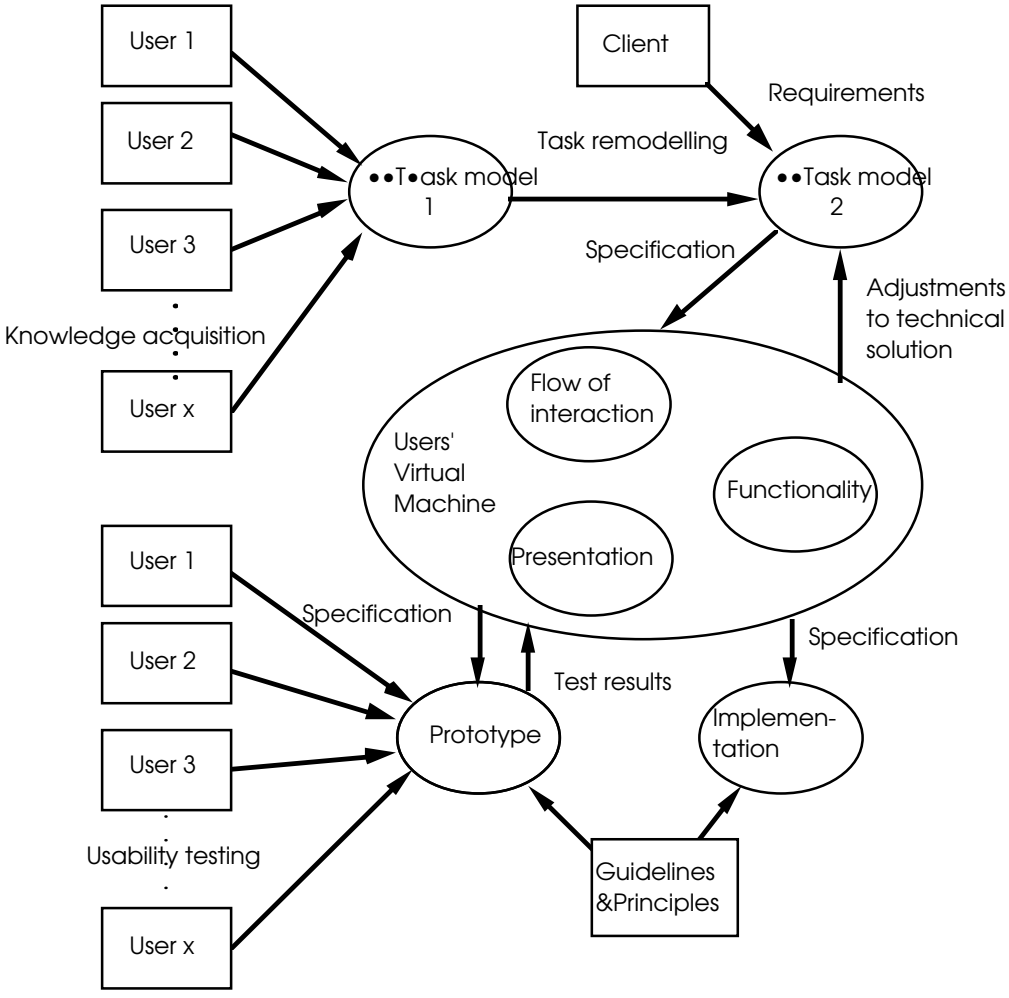

Figure 3.3: Design team, structure of activities
Source: van der Veer (1994)

Model 1: Analysing the current situation

In many cases the design of new systems is triggered by an existing task situation. Either the current situation is considered not optimal, or new technology is expected to allow improvement of current task situation.

Analysing the current situation enables the analysts to formulate design requirements, and allow evaluation of the design later on. Explicitly modelling the current situation has certainly advantages. When the task has to be modelled explicitly, the task is analysed more thorough because structured models dispose gaps in the knowledge of the analyst. Also the task model of the current situation can be used as a basis for modelling the new situation.

Model 2: Specifying the future task situation

In the second model the task situation is redesigned in such a way that it becomes possible to include technical solutions for problems and technical answers to requirements. The design decisions that lead from task model 1 to task model 2 will in actual situations be based on three different sources:

Problems identified in the first task analysis The first task analysis will reveal parts of the task structure and characteristics of the task related objects which need to be changed in order to optimise the task.

Client requirements The requirements as stated by the "client" have to be distinguished from the users problem specification as considered in the task analysis. Clients are the those who pay the design team to improve the current task situation. Clients provide requirements considering economical aspects, time constrains, and quality norms concerning the work that is done.

Technical constrains and options The technical source concerns the knowledge of the technology. On basis of this knowledge designers identify technological possibilities that can serve a solution. Technology also put constrains on the design space. Proposed solutions to problems might not be feasible with state of the art technology.

Using the design approach and the various sources does not automatically lead to model 2 and does not guarantee optimal design decisions. However the use of the sources and the task model 1, will help to structure the collection and representation of problems, alternative solutions and criteria.

Model 3: Specifying the supporting technology

The last model specifies what the system will offer the user for task delegation. While task model 2 represents the "new" task situation as a whole, the third model specifies the detailed solution in terms of technology. In this respect Tauber (1988) introduces the concept User's Virtual Machine (UVM) which indicates the knowledge that the end-user needs to have of the information technology in order to understand its competence in the domain of work. Working with technology means comparing the knowledge as determined in task model 2 with knowledge of the UVM.

The UVM can be divided into three separated parts:

- the functionality (basic activities that the user can delegate to the system),
- the language interface (the language in which the user has to express himself)

- and the presentation interface (the representation of the relevant information for the user).

All parts have to be specified explicitly.

It may be necessary to feed back the design decisions between the model 2 and the UVM, so the consequences of design decisions are kept explicit. Once the UVM is implemented in a prototype and tested a revision of both models might be necessary, because some of the d

The step wise modelling of the user interface concept as proposed by van der Veer has many advantages above a direct modelling of the user interface. The explicit modelling allow better communication between the various design teams, because both the problems and their advised solutions are explicitly modelled. Furthermore, explicit modelling allows back-tracking and evaluation of design decisions. If a consistent choice of modelling formalisms is made, this could help the design by smoothing the path between various phases of the design process. Consistency is reached when earlier models contain concepts that are elaborated or altered in the latter models.

3.3. Software designers' need for TA

Johnson and Johnson (1990) interviewed three designers in order to identify what design experience they had, what design practice they followed, what TA they performed, what their perception of the needs for TA was, and what they thought they would require in the way of tools and methods to support their use of TA in design.

All the interviewed designers followed a phased approach when designing a system. The designers studied by Johnson and Johnson all thought that they needed more detailed information about users and tasks. Two designers wanted tighter documentation of users and tasks. The other designer wanted the TA to include information about what the users needed and expected to see in the user interface and how they liked to interact with it.

The kind of information the designers wanted to know about tasks included the structure of the tasks, the frequency with which a particular procedure was carried out, the circumstances under which one procedure was used rather than another, and the inputs and outputs for each procedure.

In considering when TA might contribute to design, all the designers felt that it would have most impact during the early stages of design (analysis and requirement phase). At these stages, TA could be used to identify the functionality of the system. The user interface requirements and the TA product could also be used in the design of the interaction dialogue.

The designers that participated in the study of the Johnson's thought that TA could help them to identify:

- what users expect to have available to them at any time
- the structure and sequence of their usage of system facilities
- the names and form of representation to be given to the screen-presented objects and events
- the information that should be available in a given context (i.e. screens)

- the structure between contexts (i.e. moving between different screens)

Johnson and Johnson asked designers in what form they would find information about tasks and users most useful to them. The designers thought that the TA should support a mapping and checking operation between user tasks and proposed design. The form of the design models used can vary depending upon what design tools and methods are used, who the designer is, the subject of the design and the group the designer is in. The designers considered it important that the TA product could be mapped onto such aspects of design as functionality and dialogue design. Consequently, the result of the TA may have to be presented in various formats, one that enables checking with users and one that allow mapping on design models such as the functionality design.

3.4. Integration of Task Analysis into system design

Task Analysis should play a prominent role when identifying user requirements of the functionality and the analysis for the dynamic (dialogues) and static (screens: information that should be available) structure of the user interface. However TA should also deliver information where system analyses can be based on.

Some software design methods have their own TA modules. However these TA modules focus solely on what users should do and not on what they really do. An other problem is that they result in a task conception that can not properly be used for other purposes like user interface design. TA methods provide the analyst with a model which contains a wide range of task components. This information is not exclusively useful for the design of user interfaces. SE should benefit from the information that is enclosed in these models. Therefore it should be possible to translate the information of the TA to a format that could be used in System Analysis.

Diaper (1992) and Benyon (1992 a, 1992 b) had a lively discussion about the role of TA in the analysis phase of the design. Benyon argued that it is not possible to make new system concepts from TA because it focus on users procedures rather then on the data. In this way the new system inherits the structure from the old one. In terms of the IWS model: TA models fail to discriminate the work tasks from the enabling tasks. Also they focus on IWS structures instead of focusing on the objects and the status change of the objects.

Diaper commented that at least some TA methods deliver system independent information that can be used for the conceptualisation of new systems. Also he pointed out that the TA modules of SA methods are not user centred, because they focus on what is ought to be done and not on what is done, or on what people feel that have to be done. But this is exactly the contradiction between system analysts and HCI designers. Their focuses differ.

However I do belief that both the user interfaces and the system concept should be derived originally from one TA model. Each of the specialisms should get the appropriate information from one and the same model. Therefore there should be some synthesis between TA and SA. TA should deliver suitable information for the derivation of SA models. However this implies that TA products have to suite the SA needs. The TA objects should

contain information like objects and their relation, procedures followed by users etc.

4 Conclusions

4.1. Summary

This document shows that the number of methodologies, methods and techniques as proposed in the literature on HCI, is enormous. Those discussed in Appendix B form only a small sample of the total group. Whitefield (1994) claims that "even for the HCI researcher it is often hard to judge what the important features of any given TA method are, how methods differ and why it might be appropriate to use one in preference to another." I hope that the topics discussed in this report, contribute to a better perception in these matters

4.2 Further research

At the moment KPN has a great need for modelling methods and techniques that can create a complete overview of all activities that are executed in order to achieve certain business goals. These models can be used for design of business supporting IT. Task Analysis provide the last element in a chain, delivering models that reveal how certain elements of business activities (tasks) are actually executed by employees.

Within this graduation project, further research will be focused on one specific method that seems to be suitable. The major purpose will be to expose how a computer application can facilitate the building of task models. In a case study the usability of the tool and the method will be tested.

The TA product, the process and the tool are specified as follows:

TA product: A task modelling technique that seems to be most appropriate is the one proposed by van der Veer, Lenting and Bergevoet (1995). Their task modelling technique is object-based. It models several task components (tasks, objects, IWS structures) as objects which are related in several ways to each other. The method uses tree diagrams and templates to record the objects and their relation. The core of the task model is formed by the tree diagrams which exposes a task decomposition and task sequences from the perspective of task performers. This decomposition can be derived from various data sources like interviews observations and document studies. The method also deliver object structures which expose divers relations between objects that are relevant for the task.

The task model that is created after using the method of van der Veer et al., is easy to understand, because it uses natural language to specify the model objects and tree diagrams to display relations. It contains much task information that can be used by all parties involved in system design. Its object based nature makes it possible to use computers to support the management and building of the model. The object based formalism of the method corresponds with current trends in user interface design where object oriented design approaches become more

and more adapted.

TA Process: Although many authors describe what should be modelled, only a few also give tips for collecting and modelling the data. Very little practical guidelines exist for applying task analysis. During the the graduation project I will look for such guidelines. Eventually the method has to subscribe rules, techniques and heuristics for all stages of the Task Analysis process. However in my thesis I will focus on the first three stages; preliminary stage, the collection of data and the modelling stage.

Supporting Tools: The method proposed by van der Veer et al. can lead to extensive task models which are hard to manage and difficult to analyse. A computer tool can aid the building and analysis of the task models in several ways:

- reduction in the skill necessary to apply an TA method
- improved correctness of the method and its application
- faster, less effortful application of the method
- improving the reusability of task models
- better management of the TA product, allowing different views on the data.

The research will result in a prototype of a computer tool that supports the building of a task model by applying the method as proposed by van der Veer et al. (1995). The prototype will be evaluated in a case study. On basis of the results of this case study some recommendations for the improvement of the tool will be formulated.

Literature references

Barber C., Stanton, N.A. (1994); *Task analysis for error identification: a methodology for designing error-tolerant consumer products*; Ergonomics; vol. 37 no. 11; p. 1923-1942

Barnard, P.J. (1987); *Cognitive resources and the learning of human-computer dialogues;* In Carroll, J.M. (ed); Interfacing thought: Cognitive aspects of human-computer interaction; MIT Press

Benyon, D. (1992a); *The role of task analysis in system design*; Interacting with computers, vol. 4, no 1, p. 7-12

Benyon, D. (1992b); *Task analysis and system design: the discipline of data*; Interacting with computers, vol. 4, no. 2, p. 246-259

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

Dowell J., Long J.B. (1989): *Towards a conception for an engineering discipline of human factors*; Ergonomics; vol. 32, no. 11, p. 1513-1535

Drury, C.G. (1990); *Methods for Direct Observation of Performance*; In Wilson, J.R. and Corlett, N.J. (eds); Evaluation of Human Work, p. 35-57; London: Taylor & Francis

Diaper, D. (1989); *Task Analysis for human-computer interaction*; Ellis Horwood Limited

Diaper, D. & Addison, M (1992); *Task analysis and systems analysis for software development*; Interacting with computers, vol. 4, no. 1, p. 124-139

Eason, K. (1982); Human factors in information technology; Phys. Tech. 13, 196-201

Edmondson, W.H. & Meech, J.F. (1994); *Putting Task Analysis into Context*; SIGCHI Bulletin, vol 26, no. 4, p. 59-63

Ericsson, K.A. & Simone, H.A. (1980): *Verbal reports as Data*; Psychological Review; vol. 3, no. 87, p. 215-225

Johnson, H., Johnson P. (1989); *Integrating task analysis in to system design*; Ergonomics vol. 32 no. 11, p. 1451-1467

Johnson, P. (1992); *Human computer interaction: psychology, task analysis, and software engineering*; McGraw Hill International (UK) limited

Kieras, D., Polson, P.G. (1986); *An approach to the formal analysis of user complexity*; International Journal of Man-Machine Studies,vol. 22, p. 365-94

Kirwan B., Ainsworth, L.K. (ed.) (1992); *A Guide To Task Analysis*; Taylor & Francis

Mayhew, D.J. (1992); *Principles and guidelines in software user interface design*; Prentice-Hall, Inc. Englewood Cliffs, N.J.

Moran, T.P. (1981); *The command language grammar: a representation for the user interface of interactive computer systems*; International Journal of Man Machine Studies, vol. 15, p. 3-50

Macaulay, L., Fowler, C., Kirby, M. and Hutt, A. (1990); *USTM: a new approach to requirements specification*; Interacting with Computers; vol. 2,no. 1,p. 92-108

Scapin, D.L. & Pierret-Golbreich, C. (1990); *Towards a Method for Task Description: MAD*; In Berlinguet, L. and Berthelette, D. (eds); Work with display units 89, p. 371-380; Elsevier Science Publishers B.V. North-Holland

Sinclaire, M (1990): *Subjective Assesment*: In: Wilson J.R. & Corlett, E.N. (eds.); Evaluation of Human Work, p. 48-88; London: Taylor & Francis

Sutcliffe, A. (1989); *Task analysis, systems analysis and design*: symbiosis or synthesis; Interacting with computers; vol. 1 no. 1, p. 7-12

Tattersall C., Engelmoer, K. (1994); Task Analysis: an enhancement to system design; Research report R&D-RA-94-1180

Tauber, M.J. (1990); *ETAG: Extended Task Action Grammar - A language for the description of the user's task language;* In Diaper D, Gilmore, D., Cockton G. & Shackel B. (eds.); Interact '90, Proceedings of the IFID TC13 third International Conference on Human-Computer Interaction; p. 163-168 Elsevier Science Publishers B.V. North-Holland

Veer, van der, G.C. (1994); *Design methods for human-computer interfaces:* In: Brunnstein, K. and Raubold, E. (eds);13th World Computer Congress 94, vol. 2, p. 188-195; Elsevier Science Publishers B.V. North-Holland

Veer G.C. van der, Lenting, B.F., Bergevoet, B.A.J.(1995): *Group Task Analysis: Modeling Complexity*: submitted for publication

Whitefield, A., Hill, B. (1994); *Comparative analysis of task analysis products*; Interacting with computers, vol. 4, no. 1, p. 289-309

Whitefield, A., Esgate, A., Denley, I. & Byerley, P. (1993); *On distinguishing work tasks and enabling tasks*; Interacting with computers, vol. 5 no. 3, p. 333-347

List of Acronyms

ATOM	Analyses for Task Object Modelling
CCT	Cognitive Complex Theory
CIA	Consumentenorder Invoer Applicatie
CLG	Command Language Grammar
CTA	Cognitive Task Analysis
ETAG	Extended Task-Action Grammar
ETHICS	Effective Technical and Human Implementation of Computer-based Systems
GOMS	Goals, Operators, Methods and Selection rules
GSM	Group de travail Spécial pour les services Mobile
HCI	Human Computer Interaction
HTA	Hierarchical Task Analysis
ICS	Interacting Cognitive Subsystems
IWS	Interactive Work System
JSD	Jackson System Development
KAT	Knowledge Analyse Technique
KPN	Koninklijke PTT Nederland
KRG	Knowledge Representation Grammar
MAD	Méthode Analytique de Description
OSTA	Open System Task Analysis
SA	System Analysis
SE	System Engineering
SSADM	Structured Systems Analysis and Design Method
TA	Task Analysis
TAG	Task-Action Grammar
TAKD	Task Analysis for Knowledge Description
TDH	Task Descriptive Hierarchy
TKS	Task Knowledge Structure
TSD	Task Structure Diagrams
USTM	User Skills & Task Match
UVM	User Virtual Machine

Appendix A: Data collection techniques

Interviews

Description of technique

The interview is probably the most widely used data gathering technique. Interviews can be structured or unstructured. Structured interviews differ from unstructured interviews in that the questions and their sequence are predetermined. The unstructured interview does have its role in the early stages of information collection. However the structured interview is more commonly used for the general collection of task-based information because structured interviews offer the opportunity for more systematic collection of data.

Required resources

There are very little resources needed for this technique, namely; one interviewer, a respondent, mostly a tape recorder and a relatively isolated room where the respondent and the interviewer have privacy and feel at ease. More extensive studies using videotape are also imaginable, when the interviewee feels the need to demonstrate some part of the task. Videotaping can also serve as a check on the interview process it self. Most of the time videotaping is unnecessary or even undesirable because it can affect the conversation.

The need for interviewers to be trained is clear. Training is necessary because when using several interviewers, they have to be consistent in their approach. The training should utilise some techniques for getting the best out of the interviews. While open-ended questions can be a means to get unexpected data, they demand much insight into the domain of the task. Therefore it is important that the interviewer gains familiarity with the task situation vocabulary and the nature of the task. This can be established through the interviewer's presence within the work situation, participation in training courses and familiarisation with various aspects of the task. In this way the interviewer can also establish some credibility with the interviewees.

Advantages

- The main advantage of the technique is that it is familiar and would seem to most respondents to be a natural approach to adopt.
- The interviewer can deal with unexpected information on-line, in a better way than a more structured "off-line" approach (such as questionnaire).
- The use of structuring within the interview offers advantages in that a consistent set of questions can be asked to each individual, and if appropriate, the response data can be treated statistically.
- There is also the flexibility of asking further when some interesting points are made.

Disadvantages

- While the use of a tape recorder may have certain advantages in terms of information collection, the subsequent analysis of the responses may be very time consuming. Although the technique can be an economic one, it can also prove to be relatively costly in person-hours. However, it is perhaps not as expensive as detailed observation.
- The advantage of interviews being social, interpersonal interaction situation can also introduce disadvantages. The relationship established between the interviewer and the interviewee may cause difficulties if there are questions of perceived work status, perceived social status, or a misconception about the purpose of the interview. There could also be a problem if the knowledge which is being requested is not accessible by the interviewee from memory, for example in case of skilled operations. There is also the possibility of bias (interviewee or interviewer), or the interviewer may be told what is expected, rather than what actually happens.

References

Kirwan, B. & Ainsworth L.K. ed. (1992); A guide to task analysis; Taylor & Francis Ltd. UK

Sinclaire, M (1990): *Subjective Assessment*: In: Evaluation of Human Work, J.R. & Corlett, E.N. (eds.), p. 48-88; London: Taylor & Francis

Observational studies

Description of technique

Observational techniques are a general class of techniques whose objectives it is to obtain data by directly observing activities or behaviours. A wide range of observational techniques are in common use. Some examples; direct visual observation, remote observation (Close circuit TV) or video recording, participant observation, time-laps photography, etc.

There is a strong tendency for people to react to being observed and recorded. For this reason, one of the most important characteristics of an observational technique is the extent to which it intrudes, or appears to intrude, on operator's sense of view. The observation can occur at varying levels of "intrusion". Intrusion refers to the degree to which the observed personnel being observed should be told that they are to be observed, but the actual physical presence of the observer may affect the performance of the observed person or group. Kirwan et all. (1992) define three levels of intrusion. The first level is called "observer unobserved". With this level the observation is done mediated via close circuit television. The second level of intrusion is labelled "observer observed". In this level the observer is co-located with the operation being carried out, in that the personnel are aware of the observer's presence. The third level of intrusion is "observer participation", in which the observer actually takes part in the task alongside other operators. According to Kirwan, this type of observation can be useful when aspects of team performance are being investigated to find out how the various team members are organised and carry

out their task.

Required resources

The resources required for effective observation, depend on the quality and type of data required and on the environment of the scene that is being observed. If, for instance, audio-visual recording is used, the sophistication and quality of it will affect the cost of the observation. This is not due just to the recording equipment, but also because it will take support to make the recordings.

However, the time and costs of the analysis of the data is even more expensive and time consuming. First the data have to be reduced to a form which is useful to the analyst. This usually involves first transcribing the audio part and annotating it with observed events. Kirwan (1992) estimates that one hour of a relaxed session will take about eight hours for the basic audio transcription alone. Logging events becomes more and more time consuming when events become less easy to detect or identify , which is why it is so worthwhile ensuring that they are recorded properly in the first place.

Advantages

- Observational techniques reveal information that can not be acquired in any other way. Detailed physical task performance data can be recorded, social interactions will be captured, and any major environmental influences (noise, light, interrupts) can all be faithfully represented.
- Observation studies are ideal for pilot study since they can reveal potential behaviour patterns and influences which may not have been predicted. They allow the analyst in an exploratory study to decide what to look for, after the data have been captured.
- Observational methods can be used to identify and develop explanations for individual differences in task performance, which perhaps could only be guessed at otherwise. For instance a significant difference in performance between left and right handed operators would probably not be noticed, unless the investigator had enough insight to predict it in advance.
- Observational studies provide objective information which can be compared with information collected by another observer, or by another method.

Disadvantages

- Observational data is the rawest possible form of data. Consequently the effort which must be expended on analysis is usually considerably. It is necessary to carry out the following tasks: identify and categorise the observed events; count them; relate them to task and the system state at the time; produce a transcription of the audible content. All this must usually be done while trying to avoid the natural human tendency to prejudge the behaviour of others. Observational data can be made impartial, but its interpretation cannot.
- Observation invariably has some effect on the observed party. Even if

the operators are being watched from behind a half transparent mirror, they must be told of this and will be conscious of it. Most people eventually acclimatise to this, but waiting for this to happen is a considerable drain on resources. The analyst must therefore judge whether the type of observation selected will significantly alter the performance aspect under investigation.

- Situations which produce good or context-rich observational data are rarely the ones which precise and controlled data. This is because observable behaviour is better produced in natural task environments, rather than in the laboratory. Users of observational techniques must accept that data will tend to be incomplete and inconsistent, and certainly not usually compatible with automatic analysis
- Observational techniques can not provide any information on underlying thought processes, and so they will be of little use for highly cognitive tasks.
- The equipment needed for producing high quality observational data can be expensive and difficult to set up. This will clearly depend on such factors as the mobility required, the lighting and the noise levels. The resources for the analysis of the data can also be high.

References

Kirwan, B. & Ainsworth L.K. ed. (1992); A guide to task analysis; Taylor & Francis Ltd. UK

Drury, C.G. (1990); *Methods for Direct Observation of Performance*; In Wilson, J.R. and Corlett, N.J. (eds); Evaluation of Human Work; , p. 35-57; London: Taylor & Francis

Verbal protocols

Description of technique

Verbal protocols are verbalisations made by a person while carrying out a task. These verbalisations contain commentary about actions and their immediate perception of the reasons behind them. People are asked to think aloud while performing a task, to avoid distortion, or forgetting that could occur if the reporting were left until afterwards. Although this technique will provide information about simple actions, mostly its major aim is to obtain (direct or indirect) information on the mental processes that underlie performance which are not directly observable, such as making decisions.

Of course it is not possible to verbalise all mental processes directly. A person can only verbalise those mental activities of which information can be retrieved from the short term memory. This has the following important implications:

- The protocols have to be produced simultaneously with the task performance, because within a second they will either be passed into the long term memory or have been forgotten. Therefore if verbalisation is delayed, the individual will first have to retrieve or deduce the information. The verbalisation will be less accurate and complete, and it will interfere with the performance of the task which is being investigated

- Useful protocols can only be produced for information which is coded in a verbal form while it is in short term memory. This will preclude the use of the technique for tasks which have processed information as visual images. It is also possible that parts of the tasks are ignored because they are either seen as trivial or have become subconscious or automated due to over practice. The verbal protocol technique can not be applied successfully to such types of tasks.

Where possible the investigator should remain besides the respondent. The respondent has the feeling his or her comments are directed to a person The analyst must not put words into the respondents mouth and should restrict involvement to assisting the flow of commentary in a general way.

Required resources

Verbal protocols can be acquired fairly easily by one investigator, a respondent and a tape recorder. The sessions can be completed in little longer than it takes for the respondent to complete the task. However it will take a lot effort to transcribe and analyse the data of the verbal protocols. The literature sources differ in opinion about whether the transcription has to be made by the analyst himself or to enlist secretarial help. Some authors claim that the process of making the transcription can improve and ease the analysis afterwards, because the analyst knows the task very well by then and he can also influence the transcription by emphasising those aspects which are relevant. Therefore they claim it is not good to let someone else make the transcription. Others find making transcriptions a waste of time, and do not doubt for one moment to leave this tasks to others who can do this much faster.

The amount of time spent on the analysis of the transcriptions depends on the detail which is required and the amount of ambiguous references by the respondents that have to be resolved. Kirwan (1992) gives a rule of the thumb; 1 hour of protocol recording should take between 4 and 8 hours to analyse.

Advantages

- Verbal protocols can give accurate and detailed information about simple decision-making.
- They can be assumed to provide a basis for investigating the underlying mental process of complex tasks which can not be studied in other ways. This is why many cognitive task analysis methods use verbal protocols for acquiring data.
- The data collection itself is typically quite rapid, because very few special arrangements need to be made, on the place of acquisition. It takes little longer than the time normally spent on the tasks. Therefore fairly little effort is demanded of the respondent and his or her superiors.

Disadvantages

- Many mental processes rely on visual, auditory or even mathematical imagery for their execution. It may be very difficult to verbalise such

events.

- Verbalisation may interfere with the basic task, changing either the speed, or the method of execution. Verbal protocols may reflect the results of a cognitive process rather than the processes themselves. If further insight is needed, some further interviewing has to be done after the task is finished.
- Individuals differ in their ability to verbalise their mental processes. Quite often, people will only verbalise overt actions, which are observable anyhow. Therefore some coaching may be required before they spontaneously explain their decisions.
- The analysis and coding of the verbal protocols can be time consuming.
- Verbal protocol sessions can not always be performed on the working place itself because the verbalisation of the protocols can disturb others who are in the same room.

References

Kirwan, B. & Ainsworth L.K. ed. (1992); A guide to task analysis; Taylor & Francis Ltd. UK

Ericsson, K.A. & Simone, H.A. (1980): *Verbal reports as Data*; Psychological Review; vol. 3 no 87, p. 215-225

Appendix B: Task analysis techniques, methods & methodologies

Task Analysis Methods

ATOM **- Analysis for Task Object Modelling**

Description:

ATOM is a structured method for user interface design. It uses an object oriented approach. ATOM consists of a conceptual modelling language (a TA product) and a procedure description (a TA process). Both the language and the procedure are designed to fit into the Jackson System Development method, a method for structured system design. The ATOM procedure splits the user interface design into three parts; a user requirements stage; a conceptual modelling stage; and a user interface specification stage. Three concepts are important in the ATOM modelling language:

- Principal objects
- Actions; intentional changes in the attribute state of objects
- Relations between objects

The ATOM procedure consists of guidelines by which the description language should be obtained, and once obtained, how it might be manipulated to generate the user interface specification. ATOM is unique in that it is the only task analysis method, that is fully integrated within a software design method.

Used data:

Structured interviews with task experts (incl. transcription)

Reference:

Walsh, P.A. (1989); *Analysis for Task Object Modelling (ATOM): towards a method of integrating task analysis with Jackson System Development for user interface software design;* Diaper, D. (ed.); Task Analysis for human-computer interaction, 186-209; Ellis Horwood Limited

CTA **- Cognitive Task Analysis**

Description:

CTA is concerned with the nature of mental activity required to perform a task. It provides a framework for modelling the particular psychological structures that are involved when a user carries out a task using an interface design. By constructing this model, the designer can identify how difficult it will be for a user to use the user interface. The model is called the Interactive Cognitive

Subsystems (ICS) architecture. The problem of CTA is that it is a very complex method. Even for a psychologist, it will be a difficult undertaking to identify consistently and completely all the different subsystems and their correct associations for any given task. For someone with no psychological background it is probably even hard to understand what a particular subsystem does. Therefore a expert system is developed so the allocation of subsystems to task components will be done automatically. Such a tool is an absolute must before the ICS architecture can be used by anyone other than an expert on ICS's.

Used data:

protocol analysis

Reference:

Barnard, P.J. (1987); *Cognitive resources and the learning of human-computer dialogues;* In Carroll, J.M. (ed); Interfacing thought: Cognitive aspects of human-computer interaction; MIT Press

HTA **- Hierarchical Task Analysis**

Description:

Hierarchical Task Analysis was intended to be a general approach to task analysis and has been applied to a wide range of contexts to develop a wide range of solutions. HTA is directed at decomposing a task into necessary goals, sub tasks and procedures for achieving those goals. The strength of HTA is in its empirical content. It requires the analyst to form a detailed model of the logical structure of the task. The analysis relies heavily on overt behaviour such as the physical actions of the task performer. This makes the HTA very much analytical rather than predictive.

Used data:

Interviews, observations, questionnaires, document studies.

Reference:

Shepherd, A. (1989); *Analysis and training in information technology task*; Diaper, D. ed.; Task Analysis for human-computer interaction, 15-55; Ellis Horwood Limited

KAT **- Knowledge Analysis of Tasks**

Description:

Knowledge Analysis of Tasks is a method of task analysis that is developed from the theory of task knowledge structures (TKS). KAT uses observations, interviews and document analysis to collect data on all roles that are related to

a specific task. The data for each role are analysed in terms of TKS components. In a second stage of the analysis, generalised TKS components are identified that are common to many tasks and/or many task performances. These generalised TKS's can be compared and linked from one task to an other. The cognitive theoretical basis of KAT makes it possible to predict how people (like to) perform a specific task.

Used data:

Observations, protocol analysis, questionnaires, interviews

Reference:

Johnson P. (1989); *Supporting system design by analyzing current task knowledge*; Diaper, D. ed.; Task Analysis for human-computer interaction, 160-185; Ellis Horwood Limited

TAKD **- Task Analysis of Knowledge Description**

Description:

TAKD was originally developed for capturing training requirements for Information Technology tasks. However, soon it was recognised that it might be of interest to the design of interfaces within the field of HCI. The analysis is based on the observation of an existing task that needs to be automated or improved. TAKD identifies the knowledge requirements of IT tasks at a level of generality. From the observation actions and objects of importance for the task are analysed. From this analysis generic actions and generic objects are derived which are assumed to be at a level of description that makes them independent of the technology and task in which they are observed. The generic actions and generic objects are taken as being the primitive elements of a knowledge representation grammar (KRG). KRG's describe the knowledge that is required to perform a task, without detailing the particular task or equipment that would be used in any training course.

Used data:

Observation of (extant) task

Reference:

Diaper, D. (1989); *Task Analysis for Knowledge Descriptions (TAKD): The method and an example*; Diaper, D. (ed.); Task Analysis for human-computer interaction, p. 108-159; Ellis Horwood Limited

Task Models, Task Description Languages

CCT **- Cognitive Complexity Theory**

The complexity of an interaction is modelled by cognitive complexity theory. CCT has been developed from the GOMS approach. The aim of CCT is to model the rules that an ideal user would have to acquire to use the interactive system to perform a given task successfully. By comparing the number of rules that are required, an assessment of the complexity of the interface can be made, and by focusing on the number of new rules that are required, some estimate of the amount of learning needed is obtained.

Reference:

Kieras, D., Polson, P.G. (1986); *An approach to the formal analysis of user complexity*; International Journal of Man-Machine Studies, 22, p. 365-94

CLG **- Command Language Grammar**

CLG is a language which is developed for modelling human-computer interaction. It is particularly aimed at software engineers/designers to help them to develop a detailed design description of the user interface design at six levels of abstraction. CLG could also provide a means of communicating designs between designers and could possibly be extended to provide an evaluation of certain aspects of a design.

Reference:

Moran, T.P. (1981); The command language grammar: a representation for the user interface of interactive computer systems; International Journal of Man Machine Studies, no. 15, p. 3-50

GOMS **- Goals, Operators, Methods and Selection rules**

GOMS is an analytic technique based on a theoretical problem solving model. It makes use of a model of the human as an information-processor in its optimal form (the Model Human Processor). The MHP model is used in conjunction with a model of how a user would use the system. The GOMS modelling approach can be used to predict how much time it would take an expert user to carry out a given task using the system. In this way alternative design solutions can be compared on their efficiency.

Reference:

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

KRG **- Knowledge representation grammar**

See TAKD

Reference:

Diaper, D. (1989); *Task Analysis for Knowledge Descriptions (TAKD): The method and an example*; Diaper, D. (ed.); Task Analysis for human-computer interaction, p. 108-159; Ellis Horwood Limited

MAD - **Méthode Analytique de Description**

(Analytical description method)

MAD is an object-based description technique. It models tasks as object with various attributes such as states, goals, actions and pre- and post-conditions. Besides task-objects, the models also contain structures which shows the relation between task-objects. MAD distinguishes to kinds of relations: compositions ("Part-of") and specialisation's ("kind-of"). The compositions also contains elements called constructors. These constructors show how all task-objects, which are related to one and the same task-object, are executed. The object-based structure of MAD offers both procedural and declarative information on human task performance that can be useful for the design of user-interfaces.

Reference:

Scapin, D.L. & Pierret-Golbreich (1990); *Towards a Method for Task Description: MAD*; In Berlinguet, L. and Berthelette, D. (eds); Work with displays 89, p. 371-380; Elsevier Science Publishers B.V. North-Holland

TAG **- Task-Action Grammar**

TAG is a technique for modelling user interactions. TAG is developed from a cognitive basis of user interaction. It attempts to identify the salient features of an interaction language as perceived by the user. From these features, a competence model of an ideal user is constructed. The model can be used to assess the consistency of the interaction language.

Reference:

Payne, S.J. & Green, T.R.G. (1989); *Task-action Grammar: the model and its developments*; Diaper, D. (ed.); Task Analysis for human-computer interaction, 75-107; Ellis Horwood Limited

TKS **- Task Knowledge Structures**

The KAT analysis can be used to produce a model of the task(s) in terms of TKS. Tasks modelled in terms of TKS will involve the following components:

- *A goal structure*: This structure shows how task goals can be decomposed in subgoals. Also it shows how these goals and sub goals are related to each other for execution.
- *A procedural substructure:* This substructure is directly related to the lowest level goods of the goal substructure. It contains actions and objects and the relation between them. The relation includes sequential parallel, iterative and conditional control relations. The relations exist within the body of the procedure and determine how the actions will be executed with respect to the objects. It is possible that procedures call other procedures and that one procedure is encapsulated into an larger one.
- *A taxonomic substructure:* This substructure of TKS model represents the structure of objects and their attributes

TKS models both task knowledge that people may posses of the task and /or the knowledge range of information that is needed to perform the task successfully

Reference:

Johnson P. (1989); *Supporting system design by analyzing current task knowledge*; Diaper, D. (ed.); Task Analysis for human-computer interaction, p. 160-185; Ellis Horwood Limited

Social-Organisational methodologies for requirement specification

ETHICS **- Effective Technical and Human Implementation of Computer-based Systems**

ETHICS is a system analysis methodology which uses a social-technical approach. It places as much emphasis on user job satisfaction and good organisational design as it does on good technical design. The methodology is designed for use by designers, managers and users of new technology. The method is available in the form of a handbook that can be worked through by business managers. It has 12 main steps:

- specify work mission
- describe present work activities and needs
- consider job satisfaction
- decide what needs to be changed
- set efficiency, effectiveness and job satisfaction objectives
- consider organisational options
- reorganise
- choose computer system
- train staff
- redesign jobs
- implement
- evaluate

Reference:

Mumford (1986); designing systems for business success, the ETHICS method; Manchester Business school Publication, Manchester, UK

Open System

The Open System approach to capturing requirements is user centred, social-technical and iterative. One of the major aims of the approach is to create an 'open system', one that evolves with it's users and his tasks. It consists of five main stages:

- identification of an application area and establishment of technical and user interface design teams
- user analysis
- task analysis
- prototype creation and evolution
- implementation and evolution

This approach has been applied most successfully to knowledge based systems.

Reference:

Eason, K. (1982); Human factors in information technology; Phys. Tech. 13, 196-201

USTM - User Skills and Task Match

User Skills and Task Match is a methodology for capturing requirements of "generic" systems, so it concentrates on the very earliest stages of the product concept. USTM is specially developed for software houses which want to create a single solution for many customers. The output of the methodology is a requirement specification document which consists of six parts:

- Human requirements
- High-Level functional requirements
- Detailed functional requirements
- Requirements metrics
- Organisational and user assistance requirements
- Technological requirements and constraints

Each set is aimed at a specific set of readers and covers a distinct set of issues.

Reference:

Macaulay, L., Fowler, C., Kirby, M. and Hutt, A. (1990); *USTM: a new approach to requirements specification*; Interacting with Computers; 2, 1, 92-108