A Practical Approach for Integrating Cognitive Modelling Techniques into the Development Process

Costin Pribeanu

National Institute for Research and Development in Informatics
8-10 Averescu Avenue,
71316 Bucharest
ROMANIA
e-mail: pribeanu@u3.ici.ro

Abstract. The multi-disciplinary nature of HCI broadens the scope of software engineering in order to face the user oriented design issues. This paper aims to present a theoretical framework for integrating cognitive modelling techniques into the development process. Based on a simple example the paper demonstrates how traditional task analysis can be detailed up to the cognitive analysis of the user performance and learning with a reasonable specification effort.

Keywords. human-computer interaction, task analysis, cognitive modelling techniques

Dr. Costin Pribeanu graduated in Economic Informatics from the Academy of Economic Studies in Bucharest in 1982 and took his Ph.D. degree in 1997 from the same Academy. He is author of a number of published papers and technical reports in the area of computer graphics, human-computer interaction and GIS.

1. Introduction

In the last decade as a result of joint research projects in psychology, ergonomics, human factors and software engineering, there emerged a new field of investigation- human-computer interaction (HCI). Its aim was the design of user interfaces based on cognitive ergonomics principles and the modelling of users and of tasks they are delegating to computer.

Despite the proliferation of cognitive models and their associated analysis techniques there are still few cases of their being used in real-life projects. Carroll (1991) explained this situation as the effect of a "schismatic relation" between research in the base science and technology of user interfaces development. Haan, Veer & Vliet (1991) have tried to identify the usability problems of modelling techniques themselves. Nielsen (1994) identified an "intimidation barrier" raised by the complexity of the HCI field which discouraged developers in using others but traditional methods for the design and evaluation of user interfaces.

More recent approaches draw on developing task analysis techniques like GroupWare Task Analysis [Veer, Lenting and Bergevoet, 1996] and supporting tools like Euterpe [Welie, Veer and Eliëns, 1998a] which both originate from HCI and ethnography. Although these methods show a better integration into the software life-cycle it seems they lose the evaluative power of cognitive models as to the might-be difficulties of using or learning how to operate an interface.

The purpose of this paper is to offer a theoretical framework and make a practical approach that could help the integration of the existing cognitive modelling techniques into the development process. The paper focuses on task analysis and on the idea that in an iterative development process predictions on user performance and learning can count as evaluation criteria.

The multi - disciplinarity of HCI requires an overall perspective on the interaction process. That is why the next Section is intended as a contribution to a theoretical framework for HCI, which addresses associated disciplines and highlights typical activities such as task analysis, cognitive modelling and evaluation.

As Monk and Gilbert (1995) pointed out, a crucial problem for an effective integration of human-computer interaction into the software life cycle is that of linking requirements to the implementation through a useful and usable specification. Section 3 proposes a more detailed framework, to integrate two types of cognitive modelling techniques into the development process. The first type enables the cognitive analysis of user performance and shows how analytical evaluation based on cognitive modelling can be used as a stopping criterion in task decomposition whilst the second enables the evaluation of how easy it is to learn an interaction method.

Using cognitive analysis in real-life projects is also a question of practice. Section 4 presents a case study for task analysis. The example refers a public administration system but the situation is illustrative to many other systems. Based on a hypothesis of inadequate user performance, Sections 5 and 6 make a cognitive analysis for the estimation of execution time and ease of learning. Although simple, this practical approach demonstrates how cognitive modelling techniques are actually inserted into the development process with a reasonable specification human effort.

2. A Theoretical Framework for HCI

Human-computer interaction is an inter-disciplinary area of research and design practice. Not only have constitutive disciplines such as psychology, ergonomics, human factors and software engineering contributed to its development, but also did so other disciplines such as linguistics, sociology, ethnography, management, industrial design and typography. Such a theoretical background obviously provides a large amount of knowledge the user interface designers need understand, thus making their work much more complex.

Working in an inter-disciplinary field requires a common understanding in order to take advantage of the expertise. In order to communicate, any "intimidation barrier" should disappear. Therefore a common understanding of some basic concepts and a general framework for their integration should exist. This means to know where and how different models and techniques are to be inserted into the software life -cycle and how this inclusion could help the development process.

A well-known approach in building an inter-disciplinary framework for HCI is the task-artifact cycle [Carroll, Kellog and Rosson, 1991]. This framework draws on the idea that design is an iterative process which goes on mainly based on the emulation of previous work. People using an artifact are re-defining the task which the artifact has been created for, which in turn yields new requirements. Hence the necessity for designing a new artifact.

The need for integrating inter-disciplinary concepts into the development process points to the idea of a software life-cycle. The framework presented in Figure 1 is based on the task-artifact cycle as a paradigm for the evolution of cognitive artifacts but also highlights typical activities for HCI- like task analysis, interaction scenarios design or dialogue specification.

In order to develop human computer interfaces a task analysis has to be carried on. Task requirements are being refined through cognitive analysis that relies on some cognitive model of the user. The cognitive models simulate different interaction scenarios that are optimised with respect to ergonomic criteria. Based on the interaction scenarios, the dialogue specification is possible. The implemented artifact will re-shape the user task, and so further evaluation in the context may reveal new requirements and the cycle will restart.

This framework also highlights a third layer: the disciplines that contribute to the foundation of human computer interface design. In developing user interfaces, general ergonomic requirements should be as much satisfied as cognitive ergonomic issues should be solved. The cognitive ergonomics based on psychological constraints will thus broaden the scope of task analysis to mental processes involved in using a cognitive artifact.

As pointed out in [Pribeanu, 1997], the management plays a mediator for specific knowledge, depending on the task domain and the general knowledge required for delegating a task to the computer. Recent approaches to task analysis, say GroupWare Task Analysis [Veer, Lenting and Bergevoet, 1996] [Welie, Veer and Eliëns, 1998b] implicitly admit that the way people are performing their tasks is closely related to the organisational structure they are in.

3. A More Detailed Framework

Previous to any implementation activity, more dialogue options should be considered in order to develop the interaction scenarios. The development of interaction scenarios is done within an evaluation loop by designers who plan the optimisation of task delegation to the computer using various dialogue design options. Therefore, the general framework for interaction shown in Figure 1 should be seen as an iterative development process where the relationship between task analysis and cognitive modelling is essential.

Task analysis is a specific concern for HCI of which goal is to create cognitive artifacts for supporting the user tasks. It roughly corresponds to system analysis in software engineering, save for the target system which is human interacting with computers. The difference lies in the cognitive nature of the artifact computer [Norman, 1991] that has the capability of displaying and manipulating information in order to support a representation function. In such an instance the study of humans performing a task should also include the cognitive processes and knowledge they need to carry on that particular task.

A cognitive model is an abstract device that simulates user behaviour [Howes, 1995]. In this

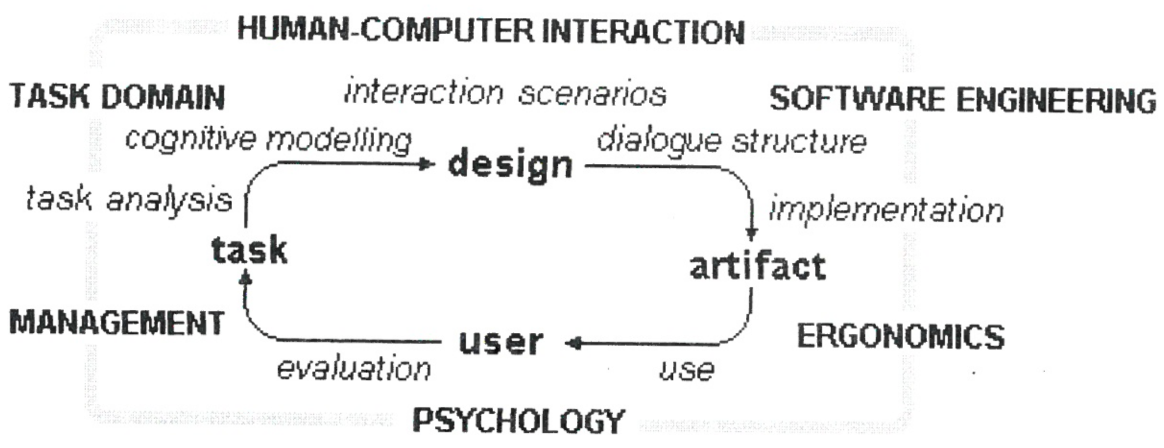


Figure 1. Theoretical Framework for HCI

respect, the model is assigned a goal, an initial state and a final state, producing a sequence of operators that describe the predicted behaviour. The representation is further used in order to make different types of predictions. Given the prediction type, Simon (1988) distinguished two categories of models:

- performance models, aiming to describe the behaviour of the user who performs a given task using a computer;
- competence models, that refer the knowledge a user has to possess in order to delegate his task to a computer.

The representation produced by performance models describes the predicted behaviour of an ideal user (who does not make errors) up to the level of physical and mental operators. The representation takes the form of a goal hierarchy. For example, the GOMS model (Goals, Operators, Methods, Selection), elaborated by Card, Moran and Newell (1983), describes the user knowledge and behaviour in terms of goals to be accomplished, operators involved, methods of interaction applied and selection rules for choosing the appropriate method.

In order to evaluate usability from the point of view of learning, an analysis of user knowledge should be carried out. Delegating tasks to computer is another task itself; therefore some representation of how this could be done is needed. In order to assess learnability, Reisner (1981) used a BNF grammar while Payne and Green (1986) employed a set grammar named TAG (Task-Action Grammar). These competence models are oriented towards the decomposition of tasks up to action level and aim to predict the ease of learning an interface by evaluating different forms of consistency in task-action mappings.

In order to take advantage of the evaluative power of cognitive models, task analysis and cognitive modelling should enter a closer relationship. The framework presented in Figure 2 integrates two types of cognitive modelling techniques into the software life- cycle. This approach views task analysis as a continuum process that refines task specification according to cognitive ergonomics criteria.

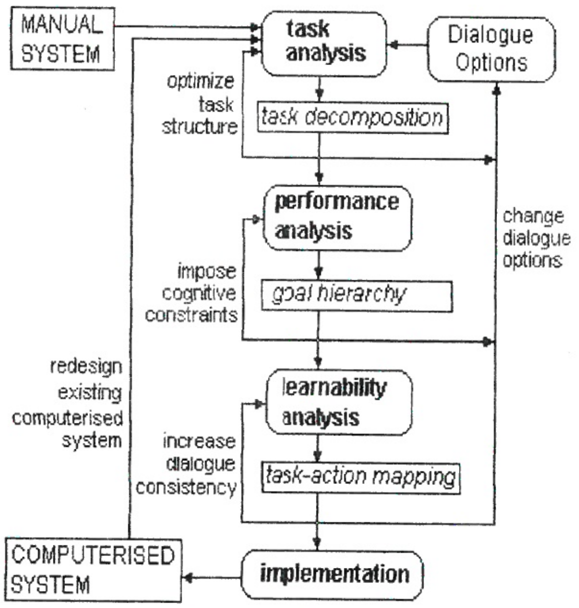


Figure 2. A Framework for Cognitive Task Analysis

The first task specification in the Figure is the result of a task decomposition. It shows how tasks are split into sub-tasks and the sequence of their being performed. Typical interaction sequences are analysed in more detail through a GOMS- like representation. This enables the optimisation of the interaction structure with respect to execution time and mental workload. If the predicted performance is not acceptable then either the dialogue options should be revised or task design should be re-considered.

Detailed dialogue design needs the mapping of tasks onto actions the user has to perform in order to operate the interface. A useful representation is a task-action grammar like BNF or TAG that, in order to improve learnability, enforces the dialogue consistency.

This methodological framework is useful for both the analysis of existing systems and the design of a new system. As the next Sections will show, to refine the representation in order to get a deeper insight into the user's task becomes possible with a reasonable specification human effort.

4. Task Analysis

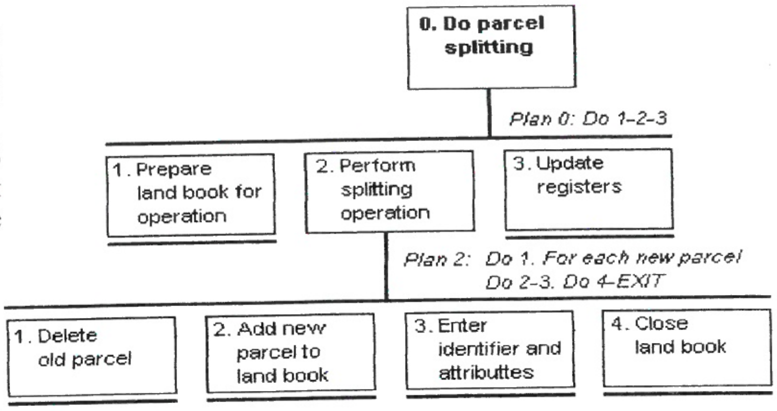
Dix et al (1993) distinguished three main approaches in task analysis, each one focussing differently: task decomposition, knowledge-based techniques and entity-relation based analysis. A well -known analysis technique is HTA (Hierarchical Task Analysis), developed by Annett and Duncan (1967), based on the hierarchical decomposition of a task into small pieces corresponding to intermediate goals.

As Sheperd (1995) pointed out, HTA is still a valid approach to task analysis. The framework described in the previous Section, seems to have HTA as the best candidate for task decomposition for at least three reasons:

- it is a simple and flexible method that does not depend on a methodological context;
- it enables the representation of a task hierarchy that could further be detailed as a goal hierarchy using an GOMS- like analysis;
- although HTA is task -oriented and sometimes user -oriented, it still maintains a strong relationship with traditional software engineering.

In order to demonstrate how the framework could actually be used, an example from a public administration system will be taken. Typical tasks there are legal registration of land parcels, the splitting or merging of parcels belonging to the same landowner, transferring the ownership of a parcel from one person to another. In order to perform the tasks other sub-tasks should be considered according to the intermediate goals identified:

- search for the client in the alphabetic register and note the land book number (if any);
- add a new client record to the alphabetic index;
- create a new land book;
- open the land book;
- delete a parcel from the land book;



0. Do parcel splitting
Plan 0: Do 1-2-3
1. Prepare land book for operation
2. Perform splitting operation
3. Update registers
Plan 2: Do 1. For each new parcel Do 2-3. Do 4-EXIT
1. Delete old parcel
2. Add new parcel to land book
3. Enter identifier and attributtes
4. Close land book

Figure 3. Task 0: Do Parcel Splitting

- add a new parcel to the land book;
- update the land book register.
- update the alphabetical owners' register.

For example, splitting of parcels into two or more plots is done according to the diagrammatic HTA representation depicted in Figure 3.

An intermediate task identification is an iterative process. In order to better realise the user task, an overview of the activity is needed. Note that some tasks may be independent tasks in a given work context while intermediate tasks in another. This aspect is important for providing a consistent structure of tasks within the task domain. A tabular representation using HTA for the same task is given below. The representation consists of small task units performed in order to achieve a goal and of plans that specify the conditions under which these tasks are to be executed.

0. Do parcel splitting
Plan 0: Do 1-2-3
1. Prepare the land book for operation
2. Perform splitting operation
3. Update registers

1 Open the land book
Plan 1: Do 1-2-3-EXIT
1. Open the territorial register
2. Identify the land book number
3. Open the land book

2 Perform splitting operation
Plan 2: Do 1. For each new parcel do 2-3. Do 4-EXIT
1. Delete old parcel
2. Add a new parcel to the land book

3. Enter identifier and attributes
4. Close the land book

3 Update registers

Plan 3: Do 1-2-3-EXIT

1. Update the land book register
2. Update owners register
3. Close the territorial register

Such a representation is device independent and it is advisable to be so for at least the first level goals. This proved useful [Knowles, 1991] since it rendered the analysis portable for both manual and computerised performance. As Knowles points out there are device independent and device dependent knowledge, each piece of knowledge being a source of cognitive complexity for the user. Given the importance of the two types of tasks in real - life systems , the usability of the whole work system has to be considered.

A minute decomposition is done according to the P x C rule [Annett and Duncan, 1967] that estimates the efficiency of proceeding on with an analysis based on the probability of a non-satisfactory performance and the cost of a non-satisfactory performance. Further decomposition of tasks requiring extra effort of analysis, is only done provided there are reasons for anticipating current performance as non -acceptable. Typical reason is an error-prone situation that could lead to serious consequences such as: inconsistent interaction methods, problems with learning the task delegation or execution time constraints. To save time, a cost-benefit estimation is useful in all these cases.

5. Prediction of the Execution Time

This Section further decomposes the task discussed in the previous one. Although using HTA makes it easy to further decompose a task, no underlying cognitive architecture is available to provide some measuring techniques of actual performance such as the estimation of execution time. So, an GOMS- like analysis technique will be used for a more detailed analysis. Then the execution time will be predicted based on the metrics provided by KLM [Card, Moran and Newell, 1983].

A related GOMS cognitive analysis technique is NGOMSL (Natural GOMS Language) developed by Kieras (1994). In this technique the goals hierarchy is mapped onto a method hierarchy that associates a method with each goal that corresponds to a "natural" structuring of the task.

Using NGOMSL makes possible a more structured and detailed representation of user behaviour and some analytical evaluation of the memory workload.

The task of splitting a parcel can be done many ways, according to dialogue options and the strategy chosen by the user. This task description in Figure 3 is based on manual registration. In a computerised system, there is no need to delete the old parcel since its data could be changed.

Suppose that the user has four functions available at parcel level: add a new parcel, delete a parcel, copy a parcel and edit parcel data. If the parcel is split into two pieces, he can rely on either the add new function, or copy function. In a NGOMSL notation, these alternatives could be represented as follows.

Method for goal: Perform parcel splitting – Add New Method

1. *Accomplish goal: Select old parcel*
2. *Accomplish goal: Edit old parcel data*
3. *Accomplish goal: Add new parcel*
4. *Return with goal accomplished*

Method for goal: Perform parcel splitting – Copy Method

1. *Accomplish goal: Select old parcel*
2. *Accomplish goal: Edit old parcel data*
3. *Accomplish goal: Copy parcel*
4. *Edit new parcel data*
5. *Return with goal accomplished*

For each goal defined, an interaction method is specified.

Method for goal: Select parcel

1. *MouseMove [parcel record]*
2. *LButtonClick*
3. *Return with goal accomplished*

Method for goal: Edit parcel data

1. *MouseMove [AddNewButton]*
2. *LButtonClick*
3. *MouseMove [parcel identifier's last character]*
4. *LButtonClick*
5. *KeyIn [/1 or /2]*
6. *KeyIn [Enter]*
7. *KeyIn [parcel area]*
8. *KeyIn [Enter]*
9. *Return with goal accomplished*

Method for goal: Add new parcel

1. *MouseMove [AddNewButton]*
2. *LButtonClick*
3. *KeyIn [placement]*
4. *KeyIn [Enter]*
5. *KeyIn [parcel identifier]*

6. *KeyIn [Enter]*
7. *KeyIn [parcel area]*
8. *KeyIn [Enter]*
9. *Return with goal accomplished*

Method for goal: Copy parcel

1. *MouseMove [CopyButton]*
2. *LButtonClick*
3. *Return with goal accomplished*

KLM (Keystroke Level Model) is a lower- level model elaborated by the same authors as GOMS's. It provides the predicted execution times for different types of operators. Thus an NGOMSL representation can be analysed from the point of view of overall execution time for a given interaction scenario and makes it possible to optimise the task. In this respect, KLM could be seen as a lower -level GOMS model in which GOMS operators like MOUSE-MOVE are further decomposed into physical and cognitive actions.

select parcel = PB[parcel record]
edit parcel data = PB[ButtonEdit] PB[identifier field] HK[/1 or /2] K[Enter] K[area]K[Enter]H[mouse]
add new parcel = PB[ButtonAddNew] HK[placement]K[Enter]K[identifier]K[Enter] K[area]K[Enter]H[mouse]
copy parcel = PB[ButtonCopy]

The operator H(home) represents hand switching between mouse and keyboard. P(position) and B(button) means positioning of mouse and pressing a mouse button. M (mental) is the operator for mental preparation of an operation (for example looking for a device on the display or searching in a displayed list for a given item) and K(Key) for pressing a key. For a more complete list of operators and associated execution times, see [Card, Moran and Newell, 1983].

In NGOMSL each statement is assumed to be a production rule for a cognitive architecture of the user and is assigned 0.05 sec in the execution time estimation. This takes into account the decision time spent for loading a method or returning to the calling one and applies for such control statements as "Method for goal", "Accomplish goal" or "Return with goal accomplished" in the representation above. Therefore, the execution time is calculated as the sum of the statement time and the operator time.

$T_{\text{Method AddNew}} = 5*0.05 = 0.25$ sec.
$T_{\text{Method Copy}} = 6*0.05 = 0.3$ sec.
$T_{\text{add new parcel}} = 10*0.05 + 1.1+0.2+0.4+14*0.28+0.4 = 6.72$ sec.
$T_{\text{select parcel}} = 4*0.05+1.1+0.2 = 1.5$ sec.
$T_{\text{edit parcel data}} = 10*0.05+1.1+0.2+1.1+0.2+0.4+9*0.28+0.4 = 6.52$ sec.
$T_{\text{copy parcel}} = 4*0.05+1.1+0.2 = 1.5$ sec.

The user typed text was assumed to be 1 character length for the placement area and 5 characters for the identifier area. After modifying the old parcel, the new parcel differs only with respect to the last character (a conventional notation that states that the new parcels are identified as OldId/1 and OldId/2).

Total Time $_{\text{Method AddNew}}$ =

$0.25+6.72+1.5+6.52 = 14.99$ sec.

Total Time $_{\text{Method Copy}}$ =

$0.3+1.5+6.52+1.5+6.52 = 16.34$ sec.

This means that the first method is faster and should be adopted. The execution time variation is due to the user's need for selecting the record in advance of editing it. However, if the execution time is not satisfactory, adding new design options, say a specialised button to perform this function, could be thought of.

6. Estimation of Ease of Learning

The models that describe and analyse the knowledge a user needs to translate a task into physical actions, employ a formal grammar in order to assess difficulties in learning to operate the interface. The basic idea of these cognitive models is to look for different forms of consistency proved to improve the learnability of the interface. Non-uniform (i.e. inconsistent) interfaces require additional rules in order to handle exceptions or special cases and make the interface more difficult to learn.

TAG [Payne and Green, 1986] relies on simple tasks defined as task units that do not include iterations or other control flow elements. The representation contains two main parts: a dictionary of simple tasks and the rule schemas. Rules are based on features that capture the regularities of the task grammar. In the tasks under consideration an example of such regularity is the positioning of the mouse to a target object like combo box button, scroll button or owner name in the list.

In order to evaluate the learnability of each alternative, this Section re-analyses the two methods in the task context. For the first alternative (Add New Method), the representation is as follows:

Dictionary of simple tasks
SelectFunction [object = AddNew Button]
SelectFunction [object = Edit Button]
SelectRecord[record= given record]
EnterText[text = given text]
ModifyText[position = given char position, text = given text]

Rule schemas

1. SelectFunction [object] → move-on[object] + Click
2. SelectRecord[record] → move-on [record] + Click
3. EnterText[text] → key-in[text] + key-in[Enter]
4. ModifyText[position, text] → move-on[target char] + Click + EnterText[text]
5. *Move-on[object= AddNew Button] → MouseMove[AddNew Button]
6. *Move-on[object= Edit Button] → MouseMove[Edit Button]
7. *Move-on[object= record] → MouseMove[record]
8. *Move-on[object= target char] → MouseMove[target char]

A comparison with the NGOMSL representation reveals that saving in the execution time between entering data for a new parcel and changing the existing data is partially due to the fact that only the last two characters of the parcel identifier need be entered. However, this requires extra learning effort, as represented by the rule schema 4 (Modify text).

The second method is slightly different.

Dictionary of simple tasks
SelectFunction [object = Edit Button]
SelectFunction [object = Copy Button]
SelectRecord[record= given record]
EnterText[text = given text]
ModifyText[position = given char position, text = given text]

Rule schemas

1. SelectFunction [object] → move-on[object] + Click
2. SelectRecord[record] → move-on [record] + Click
3. EnterText[text] → key-in[text] + key-in[Enter]
4. ModifyText[position, text] → move-on[target char] + Click + EnterText[text]
5. *Move-on[object= AddNew Button] → MouseMove[AddNew Button]
6. *Move-on[object= Copy Button] → MouseMove[Edit Button]
7. *Move-on[object= parcel record] → MouseMove[parcel record]
8. *Move-on[object= target char] → MouseMove[target char]

In order to assess learnability, Payne and Green recommend counting first the number of simple task rules schemas, and then, if the comparison reveals the same number, the total number of rules including re-write rules should be counted. Re-writing rules as 5, 6, 7 and 8 in the representation above denote "action variables" [Payne and Green, 1986] that are derived from the current goal. These rules are included in the grammar in order to describe the lexical elements of the grammar although they share the same syntactic form.

Anyway, this example shows no difference in learnability for the two methods. In both cases the user needs to know how to select a function, a given record, a given field and how to enter data. How many times a simple given task will actually be performed is of no interest for learnability. This means that for the task concerned only the execution time can be used in making the optimum design decision.

7. Conclusion

The paper presented a theoretical framework for HCI that highlighted the multi-disciplinary nature of HCI and showed the way how typical activities such as task analysis and cognitive modelling got inserted into the software life-cycle of the interaction process design. This framework was intended to give practitioners an overall perspective of the HCI.

The relationship between task analysis and cognitive modelling was investigated, as a key concept in developing more practical user interfaces and a framework that integrated cognitive modelling into the task analysis process, was proposed. Two types of cognitive models were included in this framework as complementary devices for task analysis: performance models enabling the estimation of execution time, and competence models enabling the evaluation of learnability.

A simple case study demonstrated how a task hierarchy resulting from an HTA representation could be mapped onto a goal hierarchy using NGOMSL and how the resulted specification could be evaluated for the execution time. Then the goal hierarchy was mapped onto simple tasks the user had to learn in order to delegate his task to the computer and the task – action grammar was analysed in order to assess learnability.

Although task analysis carried on within this methodological framework requires an extra specification, it is a reasonable design effort rewarded by an improvement in the final product usability. Not only will the user interface be easier to learn and faster to operate, but also will the design options and the user recommendations be documented on an analytical evaluation basis.

REFERENCES

ANNETT, J. and DUNCAN, K., **Task Analysis and Training Design**, OCCUPATIONAL PSYCHOLOGY, 41, 1967, pp.211-227.

CARD, S. K., MORAN,T. P. and NEWELL, A., **The Psychology of Human-Computer Interaction,** LAWRENCE ERLBAUM ASSOCIATES, Hillsdale, NJ, USA, 1983.

CARROLL, J.M., **Introduction:The Kittle-House Manifesto**, in J. M. Carroll (Ed.) Designing Interaction - Psychology at the Human-Computer Interface, CAMBRIDGE UNIVERSITY PRESS, 1991, pp.1-14.

CARROLL, J.M., KELLOG, W.A. and ROSSON, N.B., **The Task - Artifact Cycle**, in J.M.Carroll (Ed.) Designing Interaction - Psychology at the Human-Computer Interface, CAMBRIDGE UNIVERSITY PRESS, 1991, pp.74-102.

DIX, A., FINLAY, J., ABOWD, G. and BEALE, R., **Human-Computer Interaction**, PRENTICE HALL, 1993.

DE HAAN, G., VAN DER VEER, G.C. and VAN VLIET, J. C., **Formal Modelling Techniques in HCI**, ACTA PSYCHOLOGICA, 78, 1991, pp. 27-67.

HOWES, A., **An Introduction to Cognitive Modelling in Human-computer Interaction**, in A.Monk and N.Gilbert (Eds.) Perspectives on HCI – Diverse Approaches, ACADEMIC PRESS, London, 1995, pp.97-120.

KIERAS, D., **A Guide to GOMS Task Analysis**, Technical Report, University of Michigan, USA, 1994.

KNOWLES, C., **Can CCT Produce A Measure of System Usability?**, in D. M. Jones and R. Winder (Eds.), People and Computers IV, CAMBRIDGE UNIVERSITY PRESS, 1988, pp. 291-307.

MONK, A. and GILBERT, N., **Introduction**, in A.Monk and N.Gilbert (Eds.) Perspectives on HCI – Diverse Approaches, ACADEMIC PRESS, London, 1995, pp.5-17.

NIELSEN, J., **Guerrilla HCI: Using Discount Usability Engineering To Penetrate the Intimidation Barrier**, in R. G. Bias and D. J. Mayhew (Eds.), Cost-Justifying Usability, ACADEMIC PRESS, Boston, MA, USA, 1994.

NORMAN, D. A., **Cognitive Artifacts**, in J. M. Carroll (Ed.) Designing Interaction - Psychology at the Human-Computer Interface, CAMBRIDGE UNIVERSITY PRESS, 1991.

PAYNE , S. J. and GREEN, T.R.G., **Task Action Grammars: A Model of the Mental Representation of Task Languages**, HUMAN-COMPUTER INTERACTION, 2, 1986, pp.93-133.

PRIBEANU, C., **A Framework for the Design of Human-computer Interaction Structures in Economic Applications,** INFORMATICA ECONOMICA, Vol.1, No.2, 1997, pp.88-94 (in Romanian).

REISNER, P., **Formal Grammars and the Design of An Interactive System**, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 7, 1981, pp.229-240.

SHEPERD, A., **Task Analysis As A Framework for Examining HCI Tasks**, in A.Monk and N.Gilbert (Eds.) Perspectives on HCI – Diverse Approaches, ACADEMIC PRESS, London, 1995, pp.145-174.

SIMON, T., **Analysing the Scope of Cognitive Models in Human-computer Interaction: A Trade-off Approach**, in D. M. Jones and R.Winder (Eds.) People and Computers IV, CAMBRIDGE UNIVERSITY PRESS, 1988, pp. 35-61.

VAN DER VEER, G., LENTING, F. and BERGEVOET, A. J., **Groupware Task Analysis – Modelling Complexity**, ACTA PSYCHOLOGICA, 91, 1996, pp.297-322.

VAN WELIE, M. , VAN DER VEER, G.C. and ELIËNS, A., **Euterpe - Tool Support for Analyzing Cooperative Environments**, 9th European Conference on Cognitive Ergonomics, Limerick, Ireland, August 24-26, 1998.

VAN WELIE, M., VAN DER VEER, G.C. and ELIËNS, A., **An Ontology for Task World Models**, Proceedings of DSV-IS'98, Abingdom, UK, June 3-5, 1998.