A methodological approach to task-based design of user interfaces

Costin Pribeanu[*] and Jean Vanderdonckt [**]

*National Institute for Research and Development in Informatics,

Bucureşti, ROMÂNIA

email: pribeanu@acm.org

** Université catholique de Louvain,

Louvain-la-Neuve, BELGIUM

email: vanderdonckt@isys.ucl.ac.be

Abstract: Task-based design of user interfaces aims at supporting the design of usable interfaces by focusing on user's goals and task performance. In this approach, the task model is given the leader role among other models that constrain the design. The objective of this paper is to take a closer look at the relation between task modelling and user interface design in order to derive useful mappings between the task model and the interface model. In order to do this we need to distinguish between two levels in task modelling: a functional device independent layer as resulting from early task analysis and an operational, device-dependent layer as resulted from task design. This layered structure of the task model is exploited for grouping interaction objects in the user interface and for designing dialog units.

Key words: task analysis, user interface design, task-based design

Dr.Costin Pribeanu graduated the Economic Informatics from the Academy of Economic Studies of Bucharest in 1982 and took his Ph.D. degree in 1997 from the same Academy. He is the chair of RoCHI - theRomanian Computer-Human Interaction chapter of ACM.

Prof. Jean Vanderdonckt leads the Belgian Lab. of Computer-Human Interaction (see http://www.isys.ucl.ac.be/bchi/) at Université catholique de Louvain (Belgium) and is chair of BelCHI - the Belgian Computer-Human Interaction chapter of ACM.

1. Introduction

Design of user interfaces is constrained by several models: user model, task model application-domain model and user interface development (UIDE) model. User centred design focuses on usability and requires understanding:

- user goals – what they want to do by using the interface;
- task performance – how users are actually performing tasks.

In this paper we aim to have a closer look at the relation between task modelling and user interface design and to outline and illustrate a methodological approach to task-based design of user interfaces. Model-based approaches were mainly focused on domain model and provided with tools aiming at the computer-aided construction of the user interface starting from an abstract specification. Task-based design is user centred since it focuses on how to support the user's work with a usable interface.

In our approach, both task modelling and user interface design are iterative development processes. In task modelling we start with an initial task model that shows functional (or planning) goals – "what to do" knowledge. This model exploits the results of other activities like system analysis (data structures, processes and data flow) and user analysis (knowledge, skills and preferences). This model is developed regardless of the technology that will be used to build the interface.

Then this model is further decomposed in order to show operational goals – "how to do it" knowledge. The model is device dependent in that it depends on the available interaction objects in the UIDE. This decomposition describes how the users are accomplishing their goals with the interface.

User interface design also starts with an initial UI model. An ergonomic analysis is carried on in order to choose appropriate interaction techniques that satisfy requirements originating both from user model and

application model. Specification of interaction objects is done according to requirements coming mainly from the data model. Then grouping interaction objects according to the task model refines the interface design. Grouping leads to tune interaction object properties according to ergonomic requirements.

A more structured model could be obtained by using dialog units. In order to accommodate screen resources and various constraints of visibility and accessibility an allocation analysis is carried on. Different alternatives of placing the interaction object groups within dialog units are evaluated. When the optimal solution is decided, navigation control is added to the user interface model.

Both interaction object groups and dialog units are designed by exploiting the information provided by the task model. Compatibility of the interface with the user's tasks is an important usability criterion. Design of the user interface should mirror the goal hierarchies and the operational task structures in order to reduce the cognitive workload.

Task-based user interface design could be seen as a systematic transition between intermediate design representations. Task modelling and user interface design are not only iterative but also interleaving and this complex process needs a methodological approach in order to synchronise both activities and to ensure the completeness and consistency of design representations. Based on accurate design representations is then possible to build appropriate design tools which are able to aid the generation of the user interface.

The rest of this document is structured as follows. In next section some aspects regarding the relation between task modelling and user interface design are discussed. Then a methodological approach to task-based design is presented in section 3. Each stage in this methodology is presented within a separate sub-section. The document ends up with conclusions in section 4.

2. Related work

In a task-based approach the user interface is designed based on the information provided by the task model. Existing task models show a great diversity as regarding formalisms and depth of analysis. There are various objectives that underlie the elaboration of each task model thus featuring a certain type of representation:

- to inform design about potential problems of usability like in HTA (Annett & Duncan, 1967)
- evaluation of human performance like in GOMS (Card, Moran & Newell, 1983);
- aid design by providing a more detailed task model describing task hierarchy, object used and knowledge structures like in TKS (Johnson, Markopoulos & Johnson, 1992) or GTA (van der Veer, Lenting & Bergevoet, 1996)
- generate a prototype of a user interface like in Adept (Wilson et al, 1993).

For example, task models employed in cognitive analysis are going beyond the goal level in order to analyse cognitive workload, execution time or knowledge the user needs to posses. In this respect they relate more to user modelling. On the other hand, models aiming to support co-operative work provide formalisms able to represent how tasks are actually assigned to different roles, broadening the scope of task analysis with organisational concepts.

Recent approaches put an emphasis on the context of use, sometimes termed as the task world. In this respect, user's characteristics, people and organisations, objects they manipulate and actions they undertake - all should be taken into consideration in order to develop a usable system. As a consequence most task models have a heterogeneous conceptual framework aiming to represent tasks, roles, task decomposition, data flows, goals, actions, objects and events. The importance given to various models (environment, domain, and user) is varying according to the application type.

In a previous work (Limbourg, Pribeanu & Vanderdonckt, 2001) we showed that when using task models as prerequisites for designing UI, compatibility between the domain-task world and design knowledge is needed. In this respect the following elements are important for a task model:

- goal hierarchy, because it reflects the goal structure, not necessarily equal or similar to a task structure;
- task decomposition, because it shows the task structure and the constraints in grouping related tasks in the interface;
- temporal relationship between tasks, because they show constraints for placement of interaction objects.

Other models of the context of use provide critical elements for modelling a task:

- data flow diagram, as a complementary view that shows the functional requirements of the application in terms of information exchange;
- commands and interaction objects available, because they show the possible ways to perform an elementary task.

3. A methodological approach to task-based design

In a methodological approach, task-based user interface design is a systematic transition from task modelling to user interface design. In this approach we propose five stages:

1. choosing of appropriate interaction objects based on ergonomic criteria
2. finishing off the task design
3. grouping of interaction objects according to semantic and functional criteria
4. identifying dialog units
5. adding navigation control tasks

Figure 1 illustrates a framework for the UI design in a task-based approach and shows the contribution of related activities.

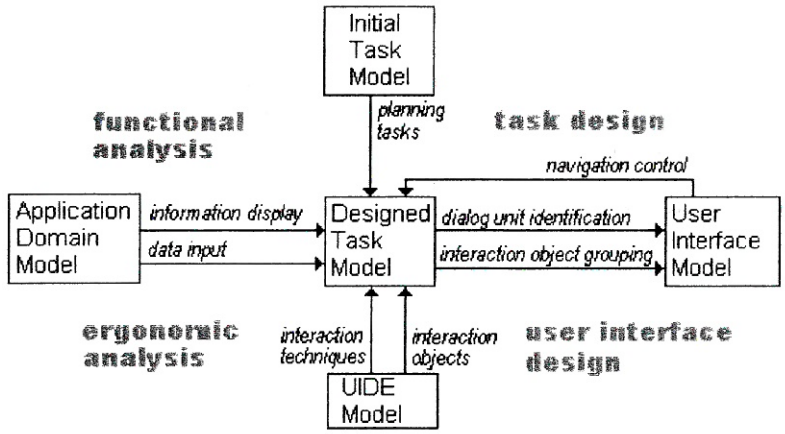

Figure 1: A framework for a task-based design of user interfaces

In this framework we distinguish between two related activities: task analysis and task design. Task analysis is an activity that is carried on early in the development cycle when technological options have not been decided. The result of this work is an *initial task model* that is reflecting basic requirements of the target domain. The lowest level tasks are associated with device-independent functional goals.

The next stage is to rephrase the high level tasks (the leaves in the task hierarchy) in terms of existing commands. Using windows based developing environment this means to decide which are the appropriate interaction objects that can support these tasks. The result of this work is *the designed task model.* The lowest level task are usually associated with interaction objects and they show how functional goals are made operational using a particular interaction technique.

The distinction between planning level and operational level is also important because it shows two activities in the development life cycle: task analysis and task design. Task analysis is usually done before choosing (or at least, regardless) the development environment. It could also be the result of a different development team (including psychologists, sociologists, ethnographers) than the one which designs the user interface.

The framework shows three pre-requisites: an initial task model, an application domain model and a user interface development (UIDE) environment model. In a task-based approach these models act as sources of information and data. In our approach the designed task model acts as a central pivot for user interface design. As it will be shown in the next sections, the relation between the task model and the user interface model is mutual in that useful information is exchanged between them in both directions.

In this approach we use a computer-aided design tool, namely the CTTE (Concur Task Tree Editor), developed by Paternò (1999), which is used to specify tasks, roles and objects as well as a task hierarchy with temporal operators. Another feature of CTTE is the attractive graphics that provide means to describe different task types like abstract, co-operative, user, interactive and application. (See Table 1)

Table 1: Graphical notation of tasks in CTT

Abstraction	Interaction	Application	User	Co-operation
				

In CTT notation operators are used to express temporal dependencies among tasks on the same level of decomposition. Table 2 summarises the temporal operators used in CTT.

Table 2: Temporal operators used in CTT

Choice	T1 [] T2
Order Independency	T1 |=| T2
Interleaving	T1 ||| T2
Synchronisation	T1 |[]| T2
Sequential composition (enabling)	T1 >> T2
Sequential composition with information passing	T1 []>> T2
Disabling	T1 [> T2
Infinite Iteration (unary operator)	T1*
Optional Execution (unary operator)	[T1]
Suspend/Resume	T1 |> T2

To avoid confusion, the temporal relationship priority in decreasing order is: unary operators, [], |=|, |||, |[]|, [> and |>, >>, []>>.

There are some restrictions in combining unary operators with others:

- T1*>>T2 is wrong , T2 is never reachable
- left and right side of the operator |> , [> and [] can not be optional

3.1 Choosing interaction objects

The user interface can be modelled in terms of interaction objects and dialog units. An interaction object (also called interactor or control) is a perceivable object of the user interface that is used to display information or to accept input from the user.

Abstract interaction objects (AIO) are supporting a generic interaction task like selecting an item from a list, choosing an option from a menu or pressing a button that triggers a transaction. An AIO is abstracting interaction capabilities of similar concrete (psychical) interaction objects (CIO) that implement the same interaction technique under different computing platforms.

Interaction objects are dynamic if they accept input from or display information to the user and static if they are only used to present or to organise the information on the screen. However both categories could be termed as interaction objects since the user is actually interacting with them either perceptually or physically.

According to these definitions, taxonomy of AIOs is given in Table 3.

Dynamic interaction objects could be used for scrolling, navigation and control.

Choosing interaction objects is the first step in our approach. It needs as prerequisites an initial task model, the available technological options – UIDE and the data model. The activities undertaken are to select appropriate interaction techniques and transform abstract interaction objects into concrete interaction objects. The result is the specification of interaction objects.

Table 3: AIO classes

Class name	Function	Typical AIOs in the class
static AIO	identification	icons, labels, group titles
	grouping	group boxes, rectangles, lines, polylines
	description	static images
dynamic AIO	scrolling	scroll arrows, scroll buttons, scroll bars
	navigation	text links, image links, information control + link, menus, navigation buttons
	information control	list boxes, combo boxes, radio buttons, check boxes, text boxes, text, status bars, progress indicators
	function control	menus, command buttons
dialog AIO	placing other AIOs	windows, dialog boxes, panels, forms, message boxes, web pages

The selection is done at abstract level in order to ensure an ergonomic approach regardless the capabilities of a particular development environment. In this respect, the data model is exploited by using selection rules based on ergonomic criteria (Bastien & Scapin, 1993) and other design knowledge. The information that could be extracted from the data model and manipulated in order to select the appropriate AIO includes:

- data type;
- data length;
- type of data processing (input, output, input / output);
- type of data domain (known, unknown);
- number of input values to choose;
- number of possible values within the domain.

An excerpt from a selection tree for the input of an alphanumeric data is given below (Vanderdonckt, 1997). In this specification L is the maximum number of characters that could be displayed on the same line and T a constant which depends on display capabilities and available screen space.

Table 4: A decision tree for AIO selection

```
-- number of input values to choose = 1
   -- domain definition = unknown
      -- data length < L: text box
      -- data length >= L: multi line text box
   -- domain definition = known:
      -- number of possible values ∈ [2, 3]: radio button
      -- number of possible values ∈ [4, 7]: radio button + group box
      -- number of possible values ∈ [7, T]: list box
      -- number of possible values ∈ [T, ∞]: drop-down box
...
```

In the case of a particular developing environment, abstract interaction objects are replaced with their corresponding concrete interaction objects. Then, properties are edited and an initial model of the user interface is obtained.

3.2 Task decomposition at operational level

A task is an activity performed by people to accomplish a certain goal. A task could be further decomposed resulting in smaller tasks corresponding to lower level goals. Task decomposition is usually represented as a tree. The first level in task decomposition shows how users are planning task performance by decomposing a task in sub-tasks and giving an ordering preference for each of them. In the initial task model goal hierarchy results from early task analysis and shows what the user really wants to do.

Depending on the technology used and the design decisions, there are several ways to make a functional goal operational. Therefore we will consider goals at both levels of task decomposition, but with different relevance. Goal hierarchies are more relevant at functional level, since they are stable across different platforms and user interface design. On contrary, task structures are more important at operational level since they are describing how the user can accomplish his goals with a given interface.

Analysing the relevance of the task from the user's point of view is important because it helps identify the knowledge he needs. This way task specification could be tuned in order to satisfy the requirements of UI design by focussing on the features that have impact on the user interface modelling.

An issue in task modelling is to identify the criteria used to decompose task in sub-task. In this respect we could distinguish the following criteria:

- functional structure – different functions of the application;
- data structure - different entities in the application domain;
- data processing - different ways the data is processed (for example iterative tasks, optional tasks, alternative tasks that depend on some parameter value);
- nature of work (cognitive, manual, input, display).

Another issue is related to the representation of task trees that show temporal relations between tasks. There are two approaches: constructors, like in MAD (Scapin & Pieret-Golbreih, 1989) or GTA (Veer, Lentig & Bergevoet, 1996) that relate task to its sub-tasks and operators like in CTT (Paternò, 1999) that relate sibling tasks. In both cases there are situations when additional tasks should be added in order to accurately express the temporal relations.

However, constructors lead to more deformations in task hierarchy since the temporal relations represent the construction mechanism of the tree. Creating an abstract task only to group tasks that have the same kind of relationship is more frequent and depends on the alternation of different temporal relations at a given decomposition level. Using operators is independent from task tree construction and leads to the creation of an additional abstract task only by exception, when priority among operators impedes the normal decomposition.

In order to illustrate the task based-approach the following example will be used. The task is to record orders submitted by a client via phone call. First the client is asked if he is an old client. If so, he is providing his id. If the search operation fails, he is asked to tell his name. If search also fails or if he is a new client then he is asked to tell his full address: street, number, postcode and city name. Second, the client is ordering the products one by one by specifying the product id (if he knows it) or the product name and the quantity ordered. The operator informs him on the total amount after each product line. Finally, the client is asked to tell the delivery date and the payment mode.

An example of goal hierarchy using the CTT notation (Paternò, 1999) and the criteria used for decomposition is given in Figure 2.

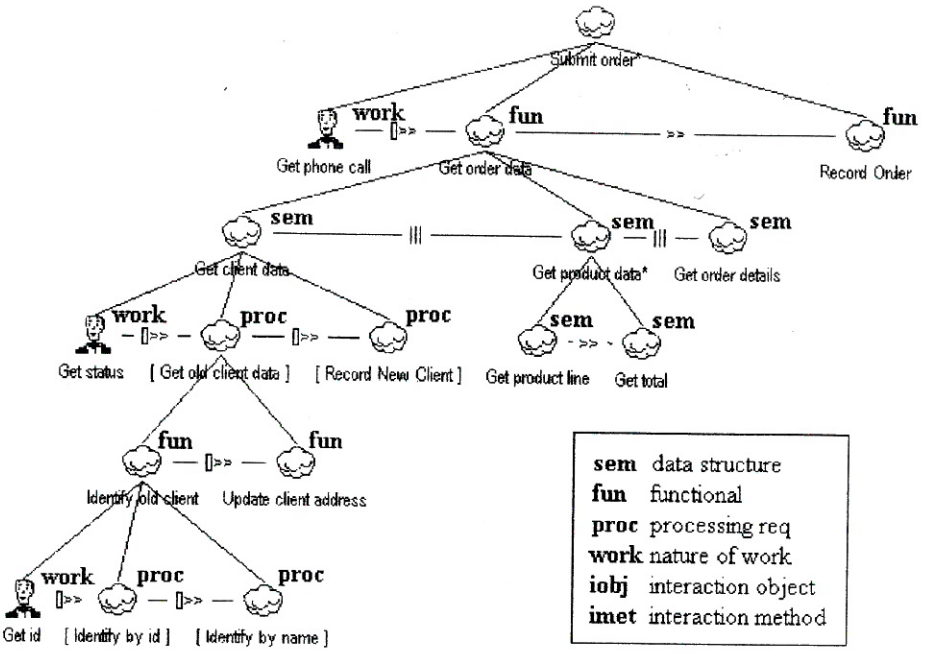

Figure 2: Goal hierarchy

The leaves in the task tree are unit tasks, defined by Card, Moran & Newell (1983) as the lower level tasks which the user really wants to perform.

Operational level is obtained by decomposing the unit tasks. According to Welie, Veer and Eliens (1998) a unit task should only be executed by performing one or more basic tasks. The relationship between unit tasks and basic tasks is important because it shows problems a user may have when he tries to accomplish his goals with a given interface. However, they give no indication how this task decomposition should be done.

Tauber (1990) defined the basic task as a task for which the system provides one single command or unit of delegation. In order to design the user interface he starts from the specification of the so-called User's Virtual Machine (UVM) which is the conceptual model of a "competent user", i.e. a user that knows how to delegate his task to the computer.

This definition of Tauber relates basic tasks to user interface objects but it leaves outside the specification of user tasks that are not related to user interface objects. Basic tasks as defined by Tauber roughly correspond to interaction tasks in CTT. Basic task classification proposed by Paternò (1999) is more complete since it comprises also user tasks and application tasks that reflect cognitive activities of the user. However, the definition proposed by Paternò – task that are not further decomposed - is ambiguous and gives no indication on the stopping criterion.

The definition of Tauber is highlighting the underlying concept for basic tasks, which is task delegation. Basic tasks are closely related to the interaction process and they are resulting from the task design activity, which is concurrently performed with user interface design activity. Their intersecting (common) work is the specification of interaction objects, which are the means, provided for the achievement of the user task.

Therefore basic tasks will be defined as the lowest level task that is using a single interaction object or a single external objects or serves a communicational goal. In this respect, the stopping criteria for task decomposition could be: an interaction object (for interaction tasks, application tasks and cognitive user tasks), an external object (for manual user tasks), and a distinct step in communication (for communication user tasks).

Figure 3 illustrates the decomposition for the unit task "Identify client by id".

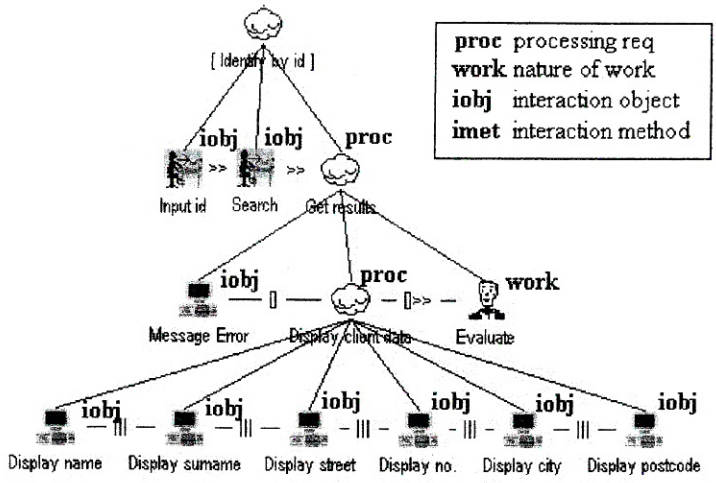

Figure 3: Task decomposition for "Identify client by id"

The decomposition at operational level is done according to the following criteria:

- processing requirements;
- external objects used;
- interaction methods used to achieve a goal, when more then one is provided;
- interaction object used.

In a similar way, Figure 4 shows task structure for the goal "Update client address". At operational level the decomposition shows interaction techniques, interaction tasks that are using interaction objects and user tasks that are related to the communication or to the manipulation of an external object.

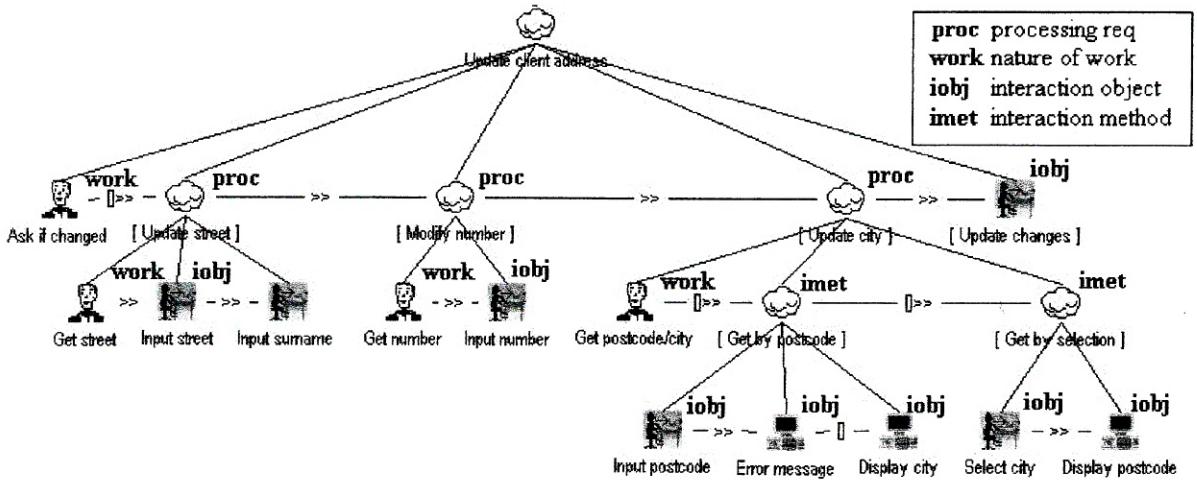

Figure 4: Task decomposition for unit tasks "Update client address"

As it could be observed, basic tasks are reached on different levels of decomposition (up to four levels in these examples). This shows that unit task structures have a variable degree of complexity.

3.3 Grouping of interaction objects

Having all interaction objects specified it is possible to build IO groups that support the achievement of all basic tasks that are accomplishing a functional (planning) goal. It is important to have this goal driven infrastructure of interaction objects in order to ensure a task-based design.

IOs could be grouped together to form a composite IO. In order to better support the mapping between the task-domain and the interface model the composite IO are restricted to one single information control object. In this respect, a combo box and the label that refers to its semantics are said to be a composite object, like in Figure 5.

Figure 5: A composite interaction object

AIOs that support related basic tasks could be further grouped together to form an AIO group. An AIO group corresponds to a complex task. The most frequent criterion used to group interaction objects is functional. In this case, a group of interaction objects has one function control IO and corresponds to a functional task unit. An example is given in Figure 6.

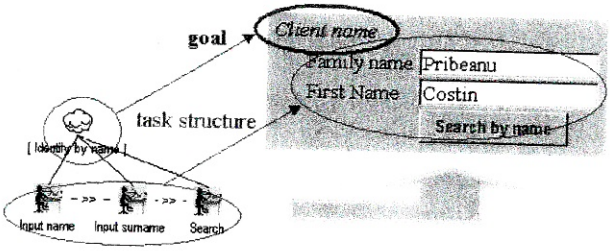

Figure 6: Interaction object group

As it could be observed, both the task structure and the goal of the unit task are mirrored in the interface. This task-based approach increases the user guidance having a favourable impact on usability.

IO groups could further be grouped together to form higher level groups. An example is given in Figure 7 where three groups are grouped together following a higher level planning goal in the goal hierarchy. In this example the goal structure for "Get old client data" is clearly represented in the interface. Moreover, the task structure is fully represented for each group, without overlapping (see Figures 3 and 4) which makes possible a semantic feedback after a successful search.

Based on the examples illustrated in Figures 6 and 7, following heuristics to exploit the task model could be derived, as shown in Table 5.

Table 5: Heuristics for grouping of interaction objects

No.	Statement
H1	Assign a static interaction object, denoting the data meaning, to each information control object.
H2	Assign a static interaction object, denoting the goal name, to each goal at unit task level.
H3	First level grouping of interaction objects should mirror the operational task structure.
H4	Higher level grouping of interaction objects should mirror the goal structure
H5	Assign a static interaction object to each higher level grouping of interaction objects, denoting the goal it represents

This way, the first level of structuring the user interface is obtained. In a task-based approach, the groups of interaction objects that mirror the last level of the goal hierarchy are the building blocks for the user interface design. According to ergonomic criteria (Bastien & Scapin, 1993) this is good for at least three reasons:

- provides user guidance, by grouping related interaction objects;
- reduces memory workload, by creating chunks of information and reducing the articulator distance needed to perform a given task;
- provides compatibility with the user tasks.

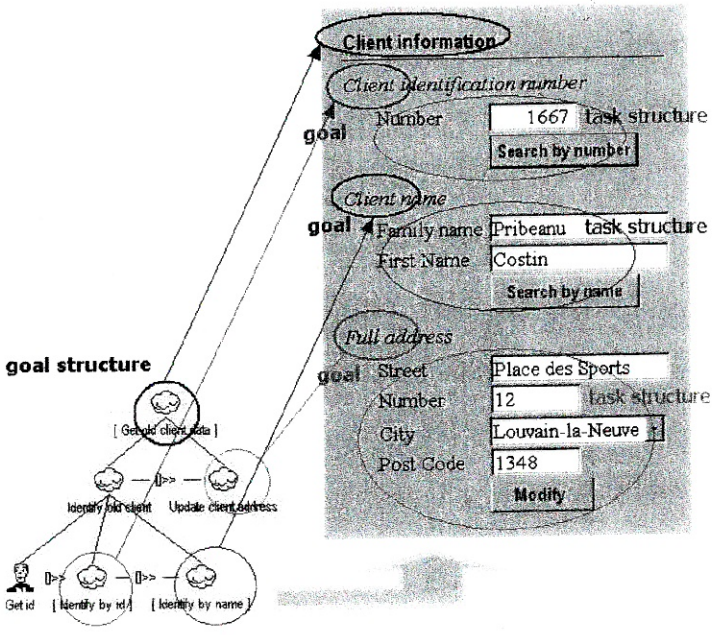

Figure 7: Higher level interaction object group

Although it is possible to imagine even higher level groups, it is recommended to avoid more then two in order to keep workload within reasonable limits. A stronger structuring of the interface could be achieved via dialog units.

3.4 Structuring the interface in dialog units

The fourth step is the identification of dialog units. A dialog unit (DU) is a collection of abstract interaction objects from which one is a dialog AIO. A dialog unit is abstracting physical window capabilities. There are two types of capabilities:

- design capabilities that enable placing other IOs in its working area ;
- interacting capabilities that enable the user to manipulate the IOs as a single unit.

The dialog unit also corresponds to a complex task but it also provides for specific tasks and or actions to be performed at this level. For example, to minimise or move the dialog unit or to scroll it's content. DUs are creating a workspace where all contained AIOs are visible.

Based on the concrete visual properties of interaction objects the area occupied by each IO group could be computed. As mentioned before, from the point of view of user interface modelling, these are the building blocks that should be grouped in dialog units. However, designing dialog units is not necessarily a bottom-up process. There are several criteria and/or constraints to apply for the structure of presentation in dialog units:

- strategy of allocation: for example to have a main dialog unit and one dialog unit for each function provided;
- different requirements regarding information processing or presentation: for example, datasheets vs. forms, forms that have a master-detail relationship; this is the case of DUs within other DU.
- built-in capabilities: for example, predefined dialog units for file management, error handling and so on;
- temporal relations between tasks: for example, a *choice* operator requires all options to be presented at a time; if this operator applies at a high level in the task hierarchy this may lead to further allocation of one dialog unit per choice;
- visibility constraints between tasks: this could be seen as a constraint (when is mandatory) or as a requirement (when is desirable); for example, having search parameters and search command visible at a time is mandatory, while displaying the address of the client after identification is desirable;
- available screen space: this requires starting dialog unit identification hierarchically, in a top-down approach.

Regarding the built-in capabilities, they are usually implemented as modal dialog units. The only problem is to include the basic tasks that enable them in the task model. The most suitable operator is *suspend / resume*.

Temporal relations are an important information provided by the task model. Analysing temporal relations between tasks is possible to derive constraints for the allocation of dialog units. We can distinguish between following situations:

- alternative tasks ([] operator): all tasks are enabled in the same time and require (at least for their first sub-tasks) to belong to the same dialog unit;
- deactivating tasks ([> operator): the disabling task (sub-task) should be within the same dialog unit as the disabled task;
- suspend resume (|> operator): the disabling sub-task should be within the same dialog unit as the suspended task;
- parallel tasks (||| operator): all tasks are enabled in the same time and require to be placed within the same dialog unit.

Tasks that are connected through the *enabling* operator could be achieved in different dialog units. Additionally, when information is passed between two tasks it is recommended that the task that will collect this information belongs to the first task (Paternò, 1999).

In order to effectively exploit temporal relations for the identification of dialog unit we need to examine the task model in a top-down direction. This way, it is possible to reduce the solution space by eliminating the solutions that are not acceptable.

There are several alternatives to structure the dialog. For example, in Figure 8 a maximal allocation is presented. Interaction objects are structured here by grouping on two levels the interaction objects.

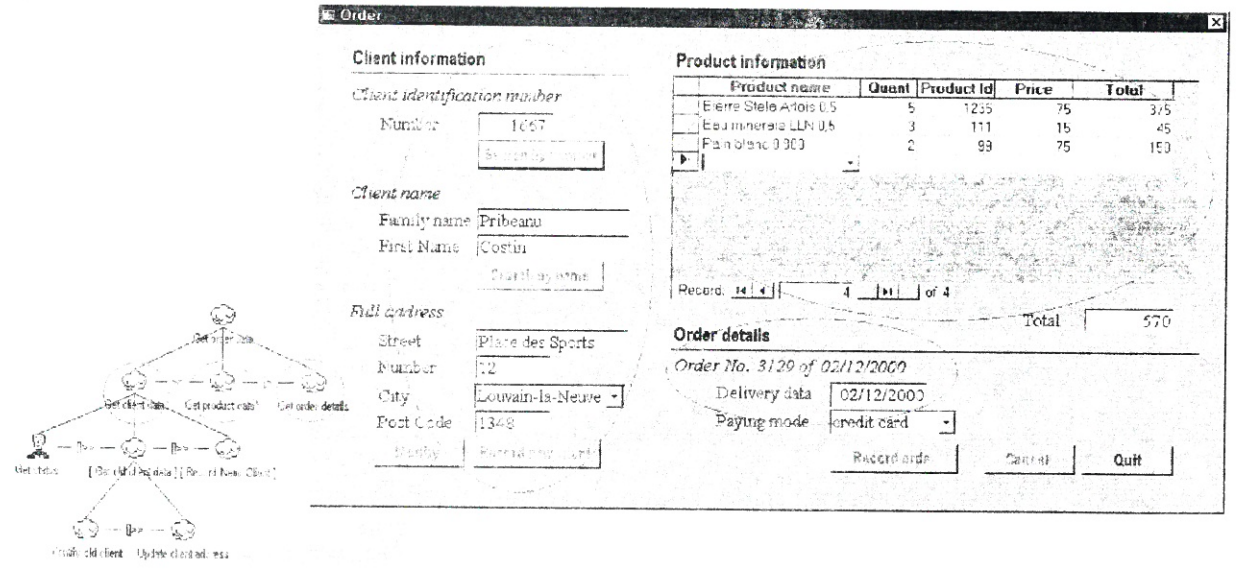

Figure 8: A maximal allocation

In this example, for each goal on the first level of goal hierarchy a group of interaction objects was designed. Each goal is marked in the interface by a static interaction object thus giving more guidance to the user. The maximal allocation is good for productivity but increases the cognitive workload of user.

Another example is given in Figure 9. In this case the first two sub-tasks were designed in two separate dialog units. The cognitive workload was significantly reduced but the user has to navigate between dialog units. Navigation control was added both in the task model and in the interface. As it could be observed, the task model changed.

The allocation was decided according to the screen area occupied by each task. Since the first two sub-tasks require more space, they were separated. Another criterion was the visibility of the whole unit task structure. Since after a successful search operation the client data is available it is good to display it.

Another possible allocation is to display the client address in a separate dialog unit (it could be done optionally, at client request). However, this holds for old client but it is not so good for new ones, when full address should be recorded.

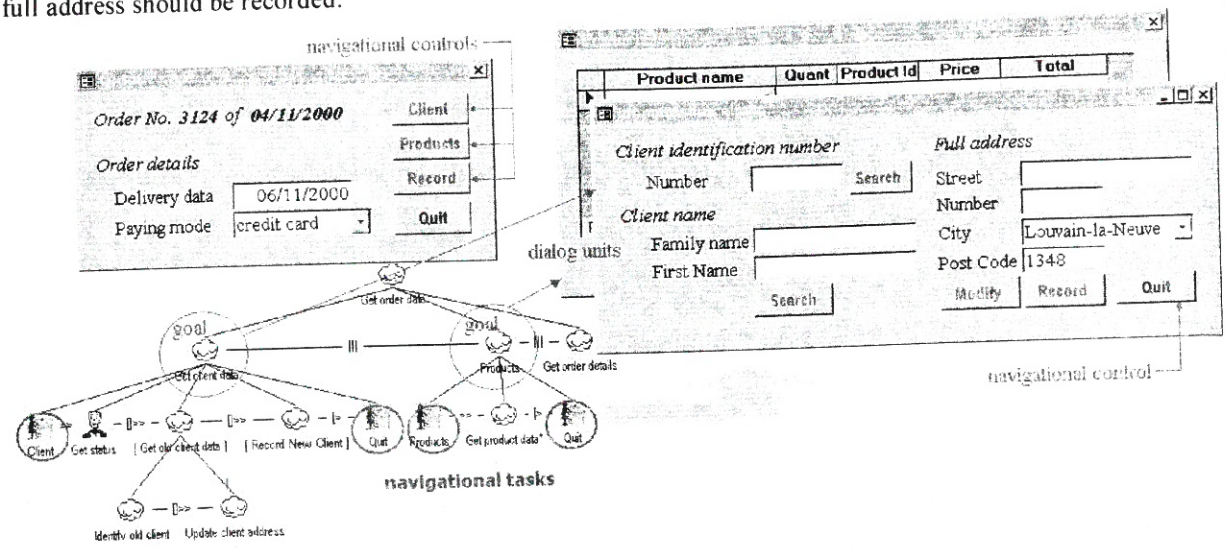

Figure 9: A task-based allocation

An important feature of dialog units is the better structuring they provide with. First, interaction objects are physically separated in a working space. Second, the goal is visible in the caption bar and only the sub-goals should be presented explicitly. This also reduces the density of interaction objects and consequently the cognitive workload.

3.5 Designing the navigation control

The last step in building the task model is to introduce the navigation control. This means to add, as appropriate, interaction objects that are needed to enable navigation between dialog units. Usually, additional interaction objects are designed only for modeless dialog boxes. However, this may change task decomposition and rise new problems since it constrains the visibility and reachability.

Usually task design (when performed) is done in a virtual space, where there is no visual constrains but only task interdependencies. The interface model splits this space in dialog units and requires a re-design of the task and an adjustment of the task model.

It is important to notice that task design is done concurrently with user interface design. Adding a new interaction object to the interface means not only to design a new task for the user but also to change the task model. According to Lim & Long (1994) task analysis does not concern solely the final task description or initial task analysis but it should also support different design perspectives: initial task situation, derivation of intermediate design description and synthesis of the new task system.

4. Conclusion

In this paper we advocated for a task-based design of user interfaces. Following our approach, task models should provide with a layered structure which follows both task analysis process and various demands to relate the task model to other models constraining and / or supporting design. According to this layered structure goals could be classified within two categories: functional goals (device independent) and operational goals (device dependent).

Operational task structure shows how the user is actually operating the interface and depends on the interaction techniques and the interaction objects selected as most appropriated for the user to accomplish his goals. Unit tasks are the lowest level of decomposition that is device Unit tasks are composed from one or more basic tasks. In turn, basic tasks are composed from user actions performed onto interaction objects and system operations that are visible in the interface.

User interface design is also an iterative work. The design of the user interface is modelled in terms of dialog units and abstract interaction objects. Abstract interaction objects could be grouped together to form AIO groups and AIO meta-groups. This layered structure is intended to give flexibility to the user interface while preserving a reasonable degree of complexity for the user.

In order to satisfy task specification the user interface should mirror as close as possible the task model. This way, user interface design could be automated using design tools that use the task model and technological options (platform, type of interface) and produce the UI components. A critical demand for such a tool is to provide means to co-ordinate at least the domain model, the task model and the user interface model.

Acknowledgement

This work has been supported by an advanced NATO fellowship and a FSR'00/QANT 01CG research project at Université catholique de Louvain.

REFERENCES

1. ANNETT, J. and DUNCAN, K., **Task Analysis and Training Design**, Occupational Psychology, *41*, 1967, pp.211-227.

2. BASTIEN, C.J.M. and SCAPIN D. (1993) **Ergonomic Criteria for the Evaluation of Human Computer Interfaces**. Technical Report No.156, INRIA.

3. CARD, S. K., MORAN,T. P. and NEWELL, A., **The Psychology of Human-Computer Interaction**, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1983.

4. JOHNSON, P., MARKOPOULOS, M. & JOHNSON,H. (1992) **Task Knowledge Structures: A Specification of user task models and interaction dialogues**. Proceedings of Interdisciplinary Workshop on Informatics and Psychology, Austria.

5. LIM, K.Y., LONG, J. **The MUSE Method for Usability Engineering**. Cambridge Series on Human-Computer Interaction. Cambridge University Press, Cambridge (1994)

6. LIMBOURG, Q., PRIBEANU, C. & VANDERDONCKT, J. (2001) "**Towards Uniformised Task Models in a Model Based Approach**". PreProceedings of 8th International Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'2001 (Glasgow, 13-15 juin 2001), Ch. Johnson (ed.), GIST Technical Report G-2001-1, Department of Computing Science, University of Glasgow, Scotland, 2001, pp.103-116.

7. NORMAN, D.A. "Cognitive artifacts". J.M.CARROLL (ed.) **Designing Interaction - Psychology at the Human-Computer Interface**. Cambridge University Press. 1991

8. PATERNÒ, F. **Model based evaluation of interactive applications**. Springer Verlag, 1999.

9. SCAPIN, D. & PIERET-GOLBREIH,C. (1989) **Towards a Method for Task Description: MAD. L.Berlinguet & D.Berthelette (Eds.) Proceedings of Work with Display Units '89**, Elsevier Science Publishers.B.V., pp.27-34.

10. TAUBER, M.J. "**ETAG: Extended Task Action Grammar - a language for the description of the user's task language**". D.Diaper et al (Eds.) Human - Computer Interaction - INTERACT 90, Elsevier, Amsterdam, 1990.

11. VANDERDONCKT, J. (1997) **Conception assisteé de la présentation d-une interface homme-machine ergonomique pour une application de gestion hautement interactive**. Doctoral disertation. Facultés Universitaire Notre-Dame de la Paix. Namur.

12. VEER, G.C. VAN DER, LENTING, B.F., BERGEVOET, B.A.J.: **GTA: GroupWare Task Analysis - Modelling Complexity**. Acta Psychologica 91, 1996, pp. 297-322

13. WELIE, M., VEER G.C. VAN DER AND ELIENS A. **An Ontology for Task World Models**, Proceedings of DSV-IS98, 3-5 June, Abingdon, UK, 1998.

14. WILSON, S., JOHNSON, P., KELLY,C., CUNNINGHAM, J. & MARKOPOULOS, M. (1993) "**Beyond Hacking: a Model Based Approach to User Interface Design**". Proceedings HCI'93, Cambridge University Press.