Encapsulating multiple perspectives in interaction specification

A. Kameas V. Gerogiannis K. Diplas P. Pintelas

Dept. of Comp. Engineering & Informatics
Univ. of Patras
Patras, Greece

Sector of Computational Mathematics & Informatics
Dept. of Mathematics
Univ. of Patras
Patras, Greece

Abstract

In this paper, a design model that regards an interactive application as a set of processes is described. Each process can be specified using the IMFG (Interactive Multi Flow Graph) model, a process model that combines Petri Net models properties with users' cognitive aspects. In this way, several perspectives of interactive applications are offered to designers; these are behavioral, informational, causal and contertual. IMFG is used as the specification model of a graphical tool that supports the design of interactive applications as the continuous refinement of users' goals, thus forcing designers to "think in users' terms". The strongest feature of IMFG is that it can be adapted to every domain of interactive applications. As an example, the model application to the design of the user interface of a highly-interactive authoring environment is briefly presented.

1 Introduction

Interaction is a highly-asynchronous process involving two agents: the user and the computer. A computer can be programmed to act in a consistent way depending on the designers' ability to implement their model of the application. Human behavior, however, is unpredictable and introduces much nondeterminism in system operation. A fundamental issue in designing interactive applications is: how to design an efficient program that would not limit users' freedom of choice [15]. This problem arises because the two "agents" involved in the design process, namely designers and (future) system users, have incompatible internal models of how the application should function. During the design of interactive applications, two processes must be carried out: the application design process (carried out by the designers) and the application-user interaction process (for which designers have to "imagine" how it will be carried out by the users). It is the embedding of the second process in the first that this paper is concerned with. More specifically, this paper presents IMFG (Interactive Multi Flow Graph), an extension of the IDFG model [7] that can be used as a graphical tool for the specification of interactive applications. In order to improve the applicability of the model in different domains of interaction, control and data information flows were separated, while user-defined scope of data had to be supported.

IMFG is a state model that is based on the formalization of Individual Token Nets (ITNs) [12], of the Petri Nets (PN) model family. PNs are a graph-theoretic tool with a strong mathematical formalism, which also have formal language properties that stem from their semantic equivalence with automata and can be used as language generators. These models are exceptionally appropriate for analyzing and modeling discrete event dynamic systems, which may be executed in parallel and exhibit synchronization and sharing phenomena [13]. In addition, PNs have many interesting properties: non-deterministic systems where concurrency and dynamic sequential dependencies exist can be naturally modeled, states and actions are equally represented, different system perspectives can be modeled, refinement and abstraction permit the modular and hierarchical representation of complex systems in a top-down and bottom-up way respectively, the system model is independent of its structure et al. Finally, PNs are a powerful communication tool: the system structure is graphically represented, while its behavior can be simulated by executing the net.

Due to these properties, PNs have been used in the past to represent, among other things, the interactive behavior of systems, but these approaches are restricted to representing some aspects of interaction (for example, X-Nets [1] represent well control and communication issues, α-Trellis [14] represents well only hypertext structures etc), and do not use PNs as an interaction specification tool at all. An interesting extension of PNs with the incorporation of objects has been presented in [2]. Petri Net Objects (PNO) use objects as the active system entities and PNs to describe their behavior. Objects can take place at several dialogs at the same time and have internally triggered activity (spontaneous activity). PNO and IMFG have many common features (such as the use of preconditions, of user-trigerred actions etc), but two major differences: IMFG adopts a process-based approach of interaction, claiming that a process element can be active only once within the same context, and uses an explicit representation of the users' context of operation to resolve conflicts (while in PNO an abstract "inner state of the dialog" is used).

0-8186-6430-4/94 $4.00

Among PNs limitations, those that mostly concern interaction are the large graphs that are usually needed to describe systems of medium complexity, the nondeterminism in state transitions and the complex conflict resolution policies required [12].

IMFG aims at helping designers regard the interaction process through the users' perspective. To this end, a new PN variation is suggested that incorporates a model of the users' cognitive capabilities (their strengths and limitations when they carry out a task). Although several models of the cognitive perspective of interaction have been proposed, no one among them represents well all the three factors that make up a model of users' cognitive capacity: short-term memory, long-term memory and perception [10]. In any case, none of these models exhibits the completeness required by an interactive applications specification tool (for example, Generalized Transition Networks combined with Production Systems [8], offer a good description of users' tasks and of computer processing, but they lack explicit representation of screen state and of the options a user has in order to accomplish a goal). To overcome PNs limitations, simplifications on the symbols used and their executional semantics have been used in the past ([16]) and conflict resolution algorithms have been proposed ([6]). In IMFG, constraints that represent the cognitive aspects of interaction are placed on the types of information and of processing elements of the model as well as on the refinement process.

A brief presentation of the background knowledge that IMFG uses will be given in the next section, together with the basic model elements and notions. As an evaluation example of the model, the application of IMFG to the modeling of the user interface of an authoring environment (GEPRIAM [11]) will be presented, followed by a brief conclusive section.

2 The IMFG model

IMFG adopts several notions from three seemingly unrelated fields: Petri Net models, process models and cognition. PNs constitute one of the more widely adopted process models that can represent well the non-deterministic, asynchronous interactive systems. In process models, interaction is viewed as a process and therefore interaction design is regarded as a software process design. The model adds a representation for the cognitive load imposed on users by using an interactive application.

2.1 Petri Net-based formalism

Interaction is an asynchronous process, not explicitly related with a strict notion of time, except in the case of design of special purpose applications (like real-time systems) and of performance modeling. Individual Token Nets (ITNs) [12] is one of the most widely used process models that is non-deterministic and does not involve any timing aspects. The main components of an ITN are *transitions* (the active system components), *places* (the passive system components used to store information) and *tokens* (which model the information that is stored in places and exchanged between transitions or modified by them).

Directed arcs are used to connect transitions to places and places to transitions. If there exists an arrow from a place p to a transition t (from t to p), then p belongs to the pre-set (post-set) of t. Each arrow may be associated with a label, which may be an expression of operations on the tokens transferred through the arrow. Correspondingly, conditions, which determine execution, may be optionally associated with each transition.

In IMFG, *actors* are used to model the processing of *tokens* and *links* to represent the information that flows among the actors; these correspond directly to the transitions and places of an ITN model. An actor is described by:

- its *input* and *output links* that make up its interface part
- its *function* included in its functional part. An actor's functionality may be inherited from existing actor templates (a concept adopted from Petri Net Objects [2]), or described using an object-oriented programming language (an approach similar to that in [1])
- a set of *rules* that make up its behavioral part. The left-hand side of each rule forms a precondition on the actors' input links, while the right-hand side part forms a postcondition on its output links.

To correctly represent all the perspectives for an interactive application links are typed, in the sense that each type holds different kind of information tokens. However, no conditions are placed on the flow of tokens.

The starting state of an ITN is represented with an *initial marking* which defines the tokens initially contained in the places of the net. A transition t is *activated* when both every input place p (p belongs in the pre-set of t) contains every token that is necessary for the substitution of t, and the condition is fulfilled. An activated transition t *occurs (fires)* nondeterministically and instantaneously two events take place: those tokens indicated by the labels of the arrows from p to t (input arrows) are removed from every input place p of t, and these tokens are added to all the output places p of t that are indicated by the labels of arrows from t to p (output arrows). A transition firing changes the marking of the ITN and is said to cause a state transition.

An actor fires when the precondition of one of the rules of its behavioral part is satisfied. The tokens of the input links specified in the rule are consumed and tokens are produced in the links specified by the postcondition of the same rule. No actor can fire unless it contains a token in its input goal link; in this case, it is added in the *actor-ready list*, where the system keeps all the actors that may fire with the next user action. Consequently, this list represents all the actions available to users towards the achievement of goal(s) pursued at the time, which can be used as a representation of state in a user-centered system. Actor firing is enacted by a token in its input event link;

with each firing the contents of the actor-ready list change.

PNs have been used to model Communicating Sequential Process-like systems, leading to a better understanding of the behavior of programs [3]. This approach, however, aims at automating the process of deriving PNs from program code, covering exactly the reverse procedure than the one presented here.

IMFG supports the *refinement* property of ITNs. During actor and link refinement, the system ensures that actors and links of the higher level can be synthesized by actors and links of the lower level in two ways:

- as a plan consisting of obligatory and probably ordered actions (AND-plan),
- by permitting the user to choose among alternative actions (OR-plan)

The other two refinement types contained in IDFG (namely, sequential plan and obligatory but independent of sequence plan) have proved of little use, and thus, have been dropped.

The perspective preserving refinement and template inheritance properties of IMFG aim at reducing the size of resulting graphs, and lead to the development of less complex application models. In IMFG, special graphs called *library actors* are used to support refinement. A library actor is a system-provided component with pre-defined behavior and functionality.

2.2 Process modeling

A process has been defined as a set of partially ordered steps intended to reach a goal. Consequently, if a user task is considered equivalent to a process, then a user action is synonymous to a process step. The three most widely adopted components of a process model are: *agent* (an actor, user or computer, who performs a process element), *role* (a set of process steps to be assigned to an agent) and *artifact* (a product created or modified by the execution of a process step).

Traditional Petri Net models represent well only the three process perspectives, namely *functional*, (represents the process steps being performed and the flow of informational entities among them), *behavioral* (represents the conditions under which these steps are performed) and *organizational* (represents the events that trigger process steps and the enactors of these events), while they constitute a weak representation means of *informational* perspective, which represents the structure and the relationships among informational entities produced or manipulated by a process. When combined, these perspectives will produce an integrated, consistent and complete model of the process described [5].

Using the link types supported by IMFG the following aspects of the interactive application can be described:

Behavioral: *condition* links are used to model flow of control and state information, defining the flow of application execution and its screen effects

Informational: *data* links are used for data and parameter passing among actors, modeling data flow during application execution

Causal: two event link types, namely *user action* and *system action* links are used to represent the events that take place in the system

Contextual: the link *goal* is used to represent the context of user actions by defining the goal to which the user plan that the actor is part of, belongs

IMFG provides multiple views of an interactive application by representing well all the four interaction process perspectives. The functional perspective can be viewed by taking into account only the functional part of the actors and by using only the links of types data, user action and system action. The behavioral perspective is represented with the behavioral actor parts and the links of types user action, system action, and condition. The organizational perspective uses user action and system action links. Finally, the informational perspective uses data links and the links refinement property. Thus, a complete model of the interaction process can be developed, enhanced with a cognitive perspective represented with goal links.

2.3 Cognitive aspects

The role of actors in interaction representation is twofold: they are the agents of the model which transform the artifacts (tokens) and at the same time they represent the processing steps of a user's task. Each actor represents an action in a users plan towards some goal.

All models of human behavior during interaction are based on the notion that the user systematically moves towards a specified goal or goals by recalling those actions that relate to a goal, forming one or more action sequences that lead to a goal, choosing one among them and executing it [9]. One of these models (GOMS [4]) divides users' knowledge into *goals* (the achievements of user actions), *operators* (the lowest-level motor and mental actions), *methods* (sequences of activities that accomplish a goal or subgoal expressed as combinations of the primitive operators) and *selection rules* (used to choose a method given certain conditions).

The cognitive load on users short-term memory is expressed by the unaccomplished goals or subgoals or other state representations that users have to keep in mind during interaction, while that on their long-term memory is usually represented with the set of actions and rules that users have to remember [10]. This load may be partially relieved if the model includes some kind of representation of screen contents.

The actor-ready list that is maintained by the system during execution of IMFG resembles the goal stack of the GOMS model. The actors in this list represent in essence all the available user actions at any moment, and consequently, can be used to define current application state. By using context-dependent actor firing rules, non-determinism of interactive applications is reduced to a level defined by user current goals. In addition, the reachability set (that is, the set

of states reachable from the current state) for any given state can be determined more easily, since only the actors in the actor-ready list have to be examined each time. Goal links are also used for conflict resolution: the actors in the same context as the last actor that fired are tried for execution first when a new event arrives. Finally, condition links can be used for the representation of screen state, enabling the designers to encompass users' perception in the design.

3 Using IMFG to represent GEPRIAM-user interaction

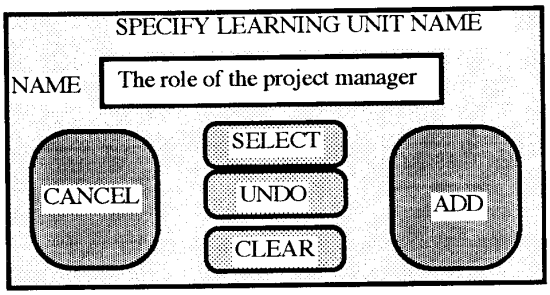
To design an interactive application using IMFG, designers have to proceed in a top-down way, from the definition of high-level goals towards the implementation of elementary actions. IMFG encompasses cognitive issues during interaction specification by forcing every actor refinement to be context-based. A consequence of this rule is that every actor must belong to a goal-leading sequence, which means that all actors have an input link of type goal. The designers will use action-modeling actors when they have reached the point where actual system operations have to be described. IMFG permits the simulation of application behavior by executing only the high-level context-modeling actors (that represent user actions towards high-level goals). In this way, a description of the behavior of the application is available at every point of the design, ensuring that user requirements will be finally met. Furthermore, IMFG promotes reusability of application parts of any size. Designers may reuse abstract computational parts, or complete user goals.

IMFG can be adapted to represent interactive applications in different domains; it suffices to define a new complete set of tokens for the goal links. An example of this important property is presented in this section, where the use of IMFG in modeling the user interface of an authoring environment is briefly discussed. In the following figures, rectangles are used to depict actors, round-edged rectangles for library actors, thick-line circles for user or system action links, dotted-line circles for goals and plain-line circles for the other links. Arrows are used to depict goal decomposition and flow of information. Note that only data tokens actually "flow" in the graph. Conditions links assume tokens from a global "pool of conditions", while event links are assigned tokens by an External Event Handling mechanism. User action and goal links are labelled for clarity (output goal links are labelled as goalOK).

GEPRIAM [11] is a highly interactive authoring environment that has been designed to impose as less cognitive load on the authors as possible, by making efficient use of screen space and by adopting a highly-structured model of the authoring process. In this case, the authoring process (which is the process of interacting with an authoring system) is separated into three distinct phases (content entry, methodology description, instructional strategy specification) each leading to the satisfaction of a macro-goal. A macro-goal is decomposed into authoring goals (a-goals), which are further decomposed into subgoals. Each a-goal is represented with an authoring tool,

Specify learning unit name	The role of the ...
Specify learning unit type	TEXT
Specify learning unit filename	CHAP1_12.TXT
Specify test type	

(a)



(b)

Figure 1: The GEPRIAM user interface elements used in the example

which corresponds to an integral context of operation. Within this context, authors may achieve authoring subgoals by using various interface widgets (such as menus, selectors, buttons etc) which support the corresponding actions. Authoring data can be input to the system by a special widget called *control*. Controls represent the context of the most elementary authoring actions.

In figure 1.a, the control menu of one of GEPRIAM tools is depicted. This tool, Reusability Base Manager, is used by the author to construct an index to the application content base by specifying the execution characteristics of learning units (one database entry per learning unit). To construct an index entry, the author must give the unit's name, type, filename that holds unit's contents, and test type, if the learning unit is of type test. To this end, the author must use four controls: Specify Learning Unit Name, Specify Learning Unit Type, Specify Learning Unit Filename and Specify Test Type. Note that, in order for the control Test Type to be available, the author must first specify "TYPE=test" using control Specify Learning Unit Type.

In figure 2.a, the IMFG that represents interaction with this menu is depicted. The overall goal (g: construct file entry) is decomposed into a set of subgoals which must all be achieved (AND-plan modeling actor): sg1 (give entry's name), sg2 (give entry's type) and sg3 (give filename). Actor C2 is further refined in figure 2.b. Note that goal sg2 is further decomposed into sg2.1 (give entry's principal type), sg2.2 (give test type, if the learning unit is a test) and sg2.3, which is

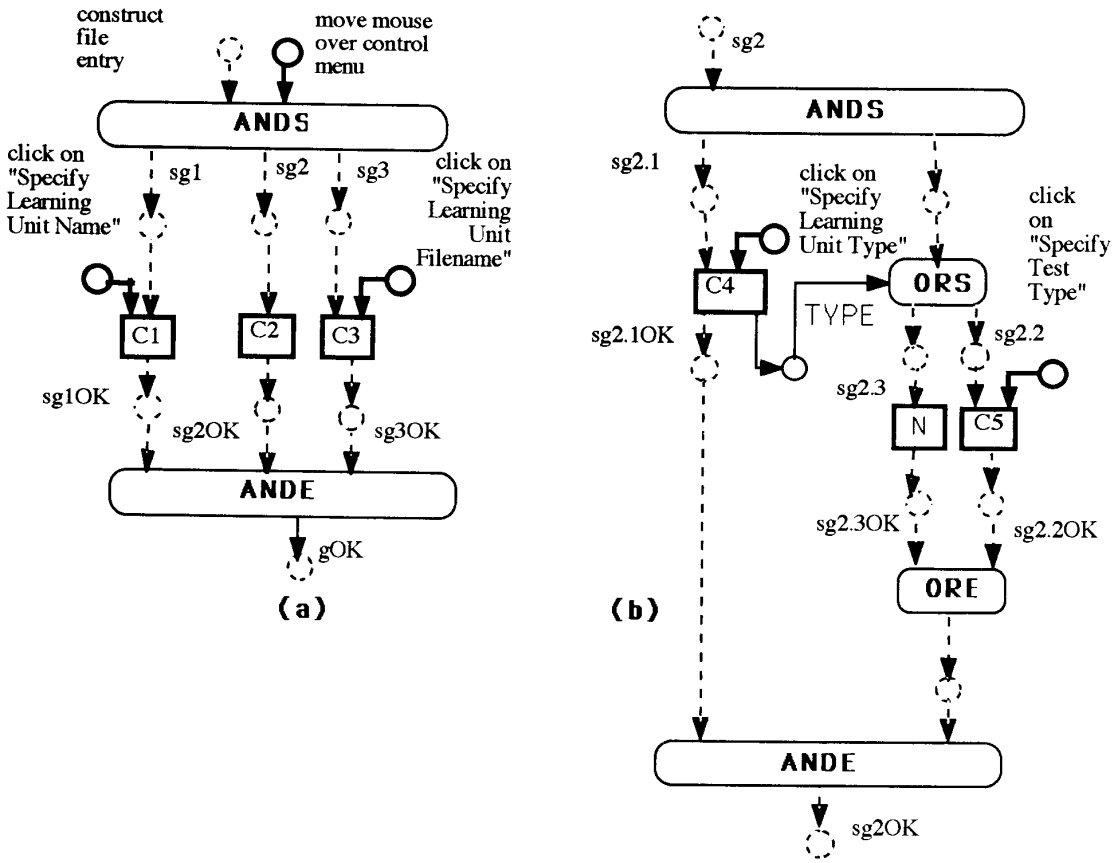


Figure 2: The IMFG that represents interaction with the control menu

a NULL subgoal. NULL subgoals, and the associated NULL actors are used for the sake of completeness and for prototyping purposes. Actor C5 does not become ready-to-fire unless actor C4 has fired and produced a token on the data link (this part of the graph is a refinement of an OR-plan modeling actor). This means that menu item 4 does not become available unless menu item 2 contains value "TEST".

In figure 1.b one of the controls of this menu (namely Specify Learning Unit Name) is depicted. By using it the authors define the unit name (by typing inside the control screen area), or open a selector to choose one (button SELECT). The author may also clear the typed name (button CLEAR), or undo the last non-definitive control operation (button UNDO). As definitive operations are considered the confirmation (button ADD) or cancellation (button CANCEL) of the contents of control screen area. Since the control closes after a definitive operation, an UNDO command does not have a context of operation and is therefore not applicable. Interaction with this control is represented with the IMFG of figure 3, which depicts a type OR-plan modeling actor that can be refined into two plans: one AND-plan and one OR-plan. The former is further refined into two OR-plan modeling actors.

These two IMFGs represent the concept of several application perspectives: by examining only the dotted links, designers can have a contextual perspective of the application, while thick lines represent a causal perspective.

4 Conclusions

In this paper, IMFG (Interactive Multi Flow Graph), a model for the specification of interactive applications has been presented. This model is a graphical design tool that encompasses several application perspectives. An application is regarded as a set of graphs composed of actors, the active model components responsible for information transformation, and links, which are used as passive entities that store and transfer information. Apart from the pure computational perspective, IMFG represents the decomposition of events that trigger actor firing, as well as the data flow among these actors. To represent this func-

tionality, IMFG is based on Individual Token Nets, one of the more general Petri Net variations. Although Petri Net models have been used in the past for the representation of interaction, there exists no model, at least to our knowledge, that incorporates users' cognitive modeling, as is the case with IMFG, which uses a link of type goal to explicitly model users goal-plan decomposition and to place user actions in context. For the representation of this aspect, IMFG is based on cognitive models of user behavior, like GOMS and Generalized Transition Networks.

IMFG resulted from the application of IDFG (Interactive Data Flow Graph) to the modeling of different interactive applications (in this paper, the representation of an interactive authoring environment was presented). IDFG proved of limited applicability to different interaction domains and many of its notions had to be generalized. Two independent types of actor refinement are supported, while link refinement is a new capability not included in IDFG. Finally, the design of an application does not point to any explicit implementation architecture, since only machine-independent sharing or communication mechanisms are specified.

A limitation of IMFG is that it does not yet include any timing aspects, which are not necessary in order to design using users' plans, but must be taken into account when designing for concurrent execution contexts and for performance modeling. In addition, the efficiency of the model has not been tested in large scale applications, where complex graphs are expected to come up. A graphical editor that, together with the extension of the model with time stamps, will enable the quantitative measuring of the efficiency of the model is currently under development. This editor will eventually be part of a complete application generator, including user interface and code generators, and full support for an IMFG reusability library.

References

[1] S. Antoniazzi, A. Balboni and W. Fornaciari, "X-Nets: a visual formalism for system specification and analysis". Proceedings of the *19th EUROMICRO Conference*, Barcelona, Spain, Sept. 6-9, 1993, pp 71-78.

[2] R. Bastide and P. Palanque, "Petri Net objects for the design, validation and prototyping of user-driven interfaces". Proceedings of *INTERACT 90*, Cambridge, UK, Aug. 27-31, 1990, pp 625-631.

[3] G. Balbo, S. Donatelli and G. Franceschinis, "Understanding parallel programming behavior through Petri Net models". *Journal of Parallel and Distributed Computing*, 15(3), 1992, pp 171-187.

[4] S. K. Card, T. P. Moran and A. Newell, *The psychology of human-computer interaction.* Lawrence-Erlbaum associates, 1983.

[5] B. Curtis, M. I. Kellner and J. Over, "Process modeling". *Communications of the ACM*, 35(9), Sept. 1992, pp 75-90.

[6] A. Javor and A. Vigh, "Conflict handling in high-level Petri Nets". *Microprocessing and microprogramming*, 39(2-5), Dec. 1993, pp 133-136.

[7] A. Kameas, S. Papadimitriou, P. Pintelas and G. Pavlides, "IDFG: an interactive applications specification model with phenomenological properties". Proceedings of the *19th EUROMICRO Conference*, Barcelona, Spain, Sept. 6-9, 1993, pp 615-624.

[8] D. E. Kieras and P. G. Polson, "An approach to the formal analysis of user complexity". *International Journal of Man-Machine Studies*, 22, 1985, pp 365-394.

[9] A. Newell and H. A. Simon, *Human problem solving.* Prentice-Hall, 1972.

[10] J. R. Olson, "Cognitive analysis of peoples use of software". In *Interfacing thought: cognitive aspects of human-computer interactiion* (J. M. Carroll, ed), MIT Press, 1987, pp 260-293.

[11] P. Pintelas, A. Kameas and M. Crampes, "Computer based tools for methodology teaching". Proceedings of the *34th ADCIS/SIGCUE Conference*, Norfolk, USA, Nov. 8-11, 1992, pp 341-355.

[12] W. Reisig, *A primer in Petri Net design.* Springer, 1992.

[13] M. Silva and J. M. Colom, "Petri Nets applied to the modeling and analysis of computer architecture problems". Proceedings of the *19th EUROMICRO Conference*, Barcelona, Spain, Sept. 6-9, 1993, pp 1-11.

[14] P. D. Stotts and R. Furuta, "Petri-Net-Based Hypertext: Document structure with browsing semantics".*ACM Transactions on Information Systems*, 7(1), 1990, pp 3-29.

[15] H. Thimbleby, *User interface design.* ACM Press, 1990.

[16] K. M. Valavanis, "On the hierarchical modeling analysis and simulation of flexible manufacturing systems with extended Petri Nets". *IEEE Transactions on Systems, Man and Cybernetics*, 20(1), 1990, pp 94-110.

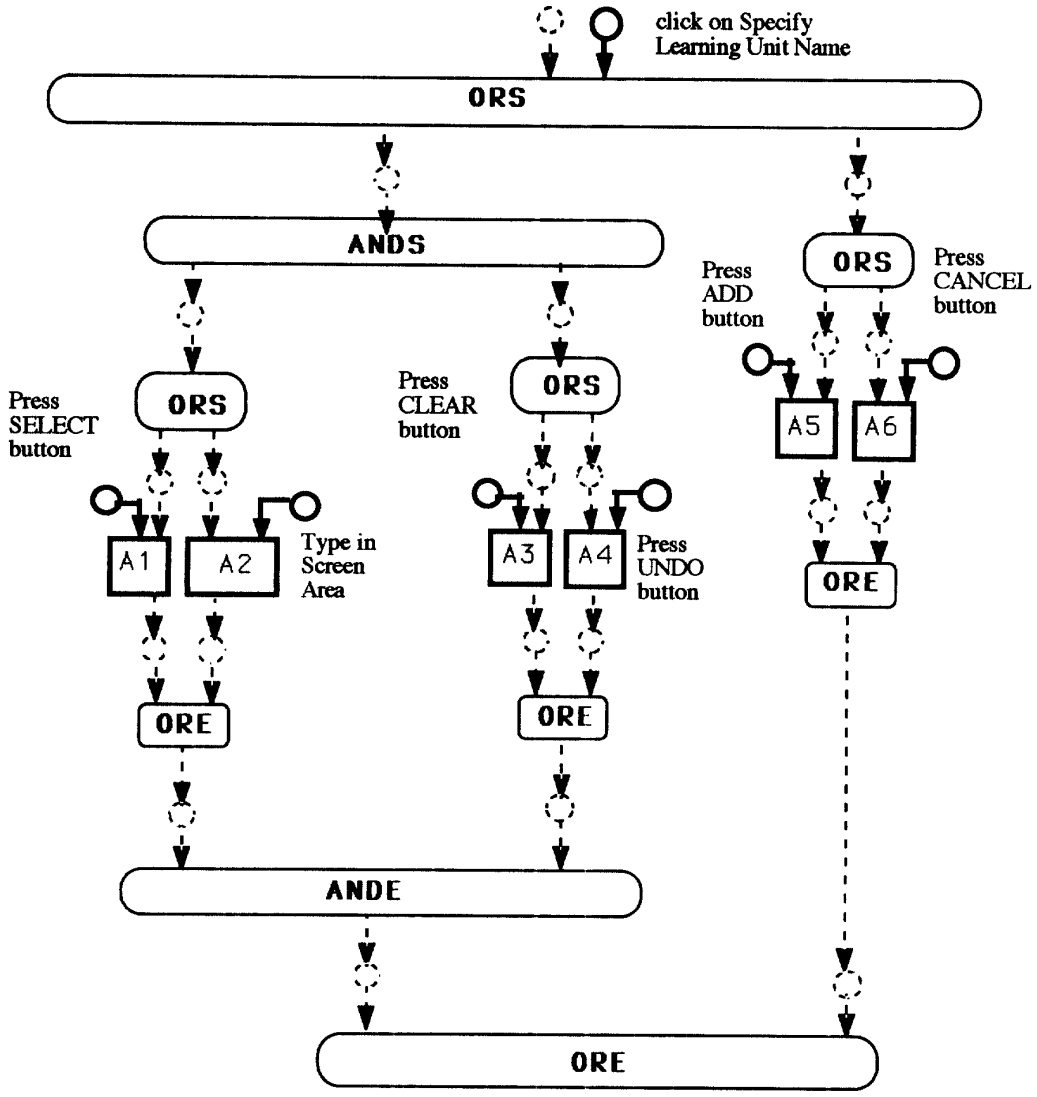


Figure 3: The IMFG that represents interaction with the control