Embedding User Models in Intelligent Interfaces

Vadim L. Stefanuk
Institute for Information Transmission Problems
Russian Academy of Science
Bolshoy Karetny pereulok, 19
101447 Moscow GSP-4, Russia
stefanuk@ippi.ac.msk.su

Abstract: The proposed Universal Intelligent Interface (UII) consists in two parts. One is a universal part, another is a problem depended part. UII, modeling user in the Frame Network User Model (FNUM), is designed as a shell system which supports a knowledge base and a number of learning agents. The system treats separately (not necessary independently) various elements of user model within the frames of a common system, which is implemented in C. Being trimmed to an application UII provides a continuos adaptation of the fine-grained user model to the concrete problem situation in process control, tutoring, intelligent support, etc.

Keywords: adaptive interface, user-centered design, learning agent, user model

Introduction

In many common programming products no User Model (UM) is used, or, better to say, just one UM is used which corresponds to an average or expected user.

Some systems support several fixed UM, one for a *novice* user, one for an *expert*, etc. [2]. Such set of UMs may create an efficient interface for the user to the rest of the system [4].

In really intelligent systems with adaptive interfaces [1] the adaptation to the user's needs and conditions should be achieved through the constant tracking and update of the current UM. For example, in [3] a simple adaptive system is described, where such a UM is being built in terms of frequency of the usage of icons in the bar of icons.

To built a high quality UM one has to observe not just a few features of user behavior [3], but many of such features [4]. We describe here a shell, allowing to capture many 'small' details of the UM which are may be updated in the system run-time. The problem dependent components in the shell are treated as attached procedures.

This 'fine-grained' UM is used then to update the properties of the current system interface, refining the interface *service*, supplied to the user. These *services* are taken from a library of available services. (See [5] for a general classification of the information, available in various interfaces.)

The presented system UII is a new implementation of the approach of the paper [6] in which Frame Network User Model (FNUM) was proposed and used to provide an intelligent support for Lisp programmers. Due to the renovation, system became more transparent and applicable in many domains, including mentioned programming support [6], tutoring systems [2], and process control [7]. A detailed description of the system UII and its core - FNUM may be found in [8].

1. General Approach

Conceptually UII consists from five separate blocks (see Fig. 1). Except the left one, all of the blocks are principally problem dependent and have to be designed separately for each application. Yet, there are certain general considerations that are applicable in many cases [4,8].

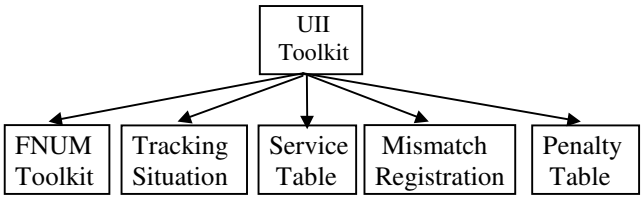

Fig. 1 Composition of System UII

Here 'FNUM Toolkit' is designed to setup, to support and update the Frame Network User Model, 'Tracking Situation' is designed to determine whether the current situation is the critical one, 'Service Table' is designed to support the information on services, recommended for the given situation and the given state of the User Model, 'Mismatch Registration' is designed to discover the reasons for updating the User Model in given situation and for current user action, 'Penalty/Reward Table' is designed to setup and to support a quantitative estimation of the current user action in order to provide reinforcement needed for adaptation of UM.

The user takes his/her action in a certain *situation*, when the user is provided with a certain service. The notions of situations and services vary from problem to problem. For example, we show a list of components of a service which are typical for industrial control interfaces [9]:

AVAILABLE SYNOPSIS
AVAILABLE DIAGRAMS
AVAILABLE PHOTOGRAPHS
POSSIBLE GOALS
POSSIBLE PLANS
POSSIBLE ACTIONS
PREFERRED INTERACTION ROLE
PREFERRED EXPLANATION STRATEGY
PREFERRED JUSTIFICATION STRATEGY
PREFERRED INFORMATIVE DETAIL
PREFERRED PRESENTATION MODE

1.2 Learning Agents

The learning (adaptation) occurs in situations which are called *critical*. A typical sequence of events might look like the following [7]:

situation→service→action→ SITUATION→SERVICE→
ACTION→ situation→service→action→ . . . .

Fig. 2 A sequence of basic triples situation-service-action.

We show the critical situations with capital letters, capitalizing the basic triple, that starts with a critical situation.

The system FNUM supports presently up to 8 learning agents, modeling up to 8 user features for each critical situation. Each agent is a finite automaton which has a number of *output states* (or its actions) and a number of *inner states*. The output states correspond to the possible states of the UM feature which is modeled by this automaton agent. Thus, for example, for the feature *skill* the output states are *weak*, *average*, *medium*, *strong*, and for the feature *fatigue* the output states are *tired, wear, fresh*.

Note, that UII (and FNUM) takes into consideration that there may be some features, like *skill*, that may vary from situation to situation [10]. Thus, a person may be very skillful in one area of a problem domain and be quite unskillful in another. From the other side, there are some features which are the same for any situation. Hence, there are two classes of automata, *local* and *global* one. Local automata are learning separately for each situation, while the global ones are learning in each of the situations.

The learning occurs if and only if the action, performed by the user, generates the penalty (or reward) for the automaton, which forces the automaton to change its inner state in accordance with one of the following two basic tactics. The *ruler* tactic used for partially ordered actions of the automaton. Otherwise, the *linear* tactic is used. Details are in paper [8].

The corresponding reinforcement is generated by the block Penalty/Reward through the Mismatch registration block which discoveries some inadequacies in user actions.

1.3 Mismatch Registration

The "Mismatch Registration' is designed to discover the reasons for updating the User Model in given situation and for current user action. When a mismatch is encountered the block sends a penalty-or-reward back to the system FNUM via Penalty/Reward block.

A mismatch may be detected in several different ways, the most typical cases are being shown in Fig. 3

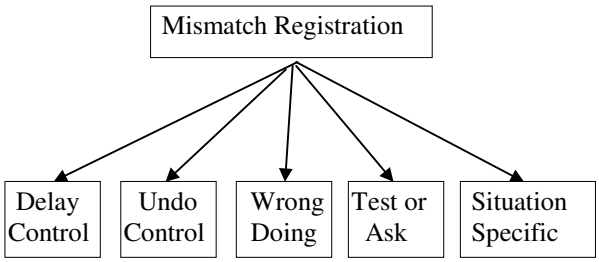

Fig. 3 The Mismatch typical cases

The block may be used both in an on-line and off-line mode for filling a column in the Penalty Table, i.e. to make a decision about the penalty value. The on-line mode of operation is the basic one. However the block itself is attached by an expert at the installation phase.

1.4 Penalty/Reward Function

The user action is a result of decision taken by the user, which is based (mainly) on the previously occurred events (see Fig. 2), the most recent ones are usually taken into account more seriously than things of the past. Depending on the action and taking into account given situation, given service, and given UM state, a reinforcement is calculated which is used by the FNUM Toolki to update current user model.

The Penalty/Reword generates a binary vector. The block receives on its input various values stored in FNUM and returns an estimation of the action, performed by the user. Some attached procedures are used for this purpose. A library of such procedures (some of them are shown in fig. 3) is to be supplied with the UII.

1.5 An Illustration: Attached Procedure For Supervised Control

In this case [9] a person is to control the plant with the purpose to provide the optimal output behavior. The system under control is an open loop system (see Fig. 4).

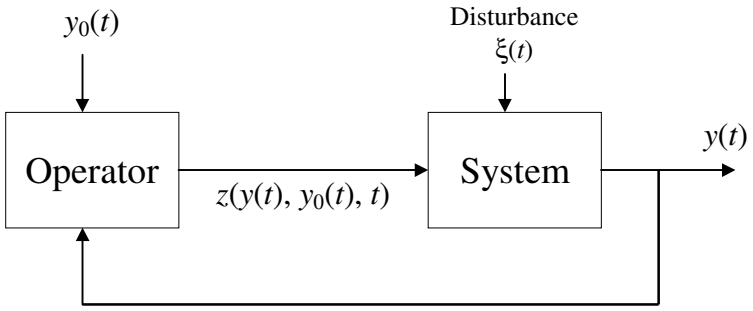

Fig. 4 Open loop control

The goal is to achieve the minimum of the expression $\frac{1}{T}\sum_{1}^{T}\left|y(t)-y_0(t)\right|$ over the time interval T. (The disturbance $\xi(t)$ is considered to be limited in its power to allow a successful control.)

It is assumed that the Operator knows partially a reverse characteristic of the System to the extent that he/she knows the sign of the derivative of y over z in the absence of the disturbance. Hence he knows what "direction" the value of z should be changed, given the $y(t)$ and $y_0(t)$, $t = 1,2,...$

The quality of the Operator decision and the resulting control sequence may not be evaluated precisely, otherwise this function will be taken from the user and given to an automatic controller. Nevertheless, one is able to make some general indirect conclusions on the decision quality. One such conclusion may be drawn from the observation of the time spent before decision is taken and the corresponding action is performed.

Indeed, the user is receiving the value of $y(t)$ at the moment of t. However, his/her reaction may fall within an interval $[\Delta_-, \Delta_+]$. The meaning of positive time lag Δ_+ is rather obvious - it is the delay in reaction, that in accordance with an operation instruction should not exceed a

certain limit - Δ_{+limit}. Theoretically, the smaller the value of Δ_+, the better. The actual amount of the Δ_+ may be used to judge on the skill level of the operator and his state of fatigue, etc.

The following chart could be used to estimate the value of delay:

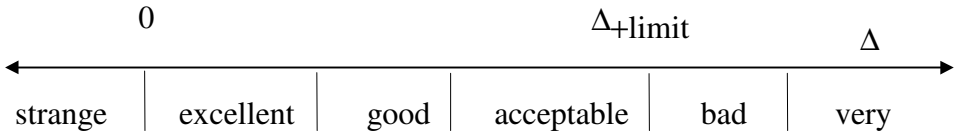

Fig. 4 Estimation of time delay

The Mismatch Registration block should be able to trace the Δ in order to adjust the user model supported by FNUM.

2. Installation of UII: Embedding User Model

For installation of UII for a given application an installation module is used. It provides a necessary support for an Expert assisting him/her in collection and organization of the following information:

- To design a set of standard (default) frame-prototypes of user-frames for the users of various class, skill, and fatigue and other features to be used in the given application of FNUM.
- To create a concrete exemplar of user frame for each users of in the given application based on the previously defined set of fame-prototypes.
- To define a list of all possible critical situations.
- To define a table relating the current critical situation to the current recommended service depending on the current FNUM state.
- To define a table relating the current critical situation, current recommended service, the current user action to the current penalty/reward value, or the current penalty/reward attached procedure, to be used in FNUM to update the current user model.

To choose the parameters of learning agents used in FNUM to correct the current user model in the given application.

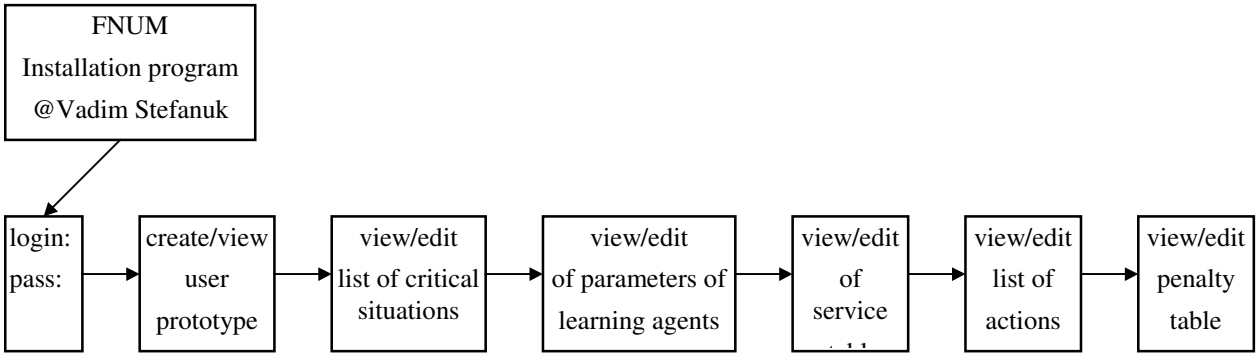

Fig. 5 The sequence of windows presented to an expert

In all cases the Expert is working with a menu type of information.

Conclusion

The proposed fine-grained adaptive interface is flexible enough to take into account the variations of the user behavior in different areas of the global task space.

The separation of the problem independent adaptation part from problem dependent blocks simplifies the design of the adaptive interface for a concrete domain. Our preliminary study shows also that the problem dependent blocks are also may be universal enough. The creation of a library of such blocks is our next immediate goal [4, 8].

Acknowledgments

The research was partially supported by Russian Fund for Fundamental Research, project RFFI-96-01-00334.

References

[1] Browne, D., Totterdel, P., Norman, M. Adaptive User Interfaces, Academic Press, London, 1990.

[2] Brusilovsky P. Program Visualization as a Debugging Tool for Novices, in Adjunct Proc. INTERCHI'93 Conference on Human Factors in Computing Systems (Amsterdam, The Netherlands, April 24-29, 1993), ACM Press, pp. 29-30.

[3] Debevc M. Adaptive bar, in Adjunct Proc. INTERCHI'93 Conference on Human Factors in Computing Systems (Amsterdam, The Netherlands, April 24-29, 1993), ACM Press, pp. 117-118.

[4] Rasmussen J. Information Processing in Human-Machine Interaction. An Approach to Cognitive Engineering. North-Holland, New York, 1986, 215P.

[5] Schlueter K., Mantei M. Formalizing User Interface Requirements, in Adjunct Proc. INTERCHI'93 Conference on Human Factors in Computing Systems (Amsterdam, The Netherlands, April 24-29, 1993), ACM Press, pp. 41-42.

[6] Stefanuk V.L. Towards intelligent operating system: flexible Lisp environment, in VTT Symposium on man- machine interface, ESPOO, Technical Research Centre of Finland, 1987, pp.117-127.

[7] Stefanuk V.L. Applications Of Non-Homogeneous Learning Automata Networks, in Engineering Applications of Artificial Neural Networks, Bulsari A. and Kallio S., Eds., Finnish Artificial Intelligence Society, Helsinki, Finland, 1995, pp. 453-456.

[8] Stefanuk V.L. A Universal Intelligent Interface, in Proceedings of JCKBSE'96, Second Joint Conference on Knowledge-Based Software Engineering, Sozopol, Bulgaria, September 21-22, 1996, pp.64-72.

[9] E.A.Averbukh, G.Johannsen, F.Mletzko, R.von Passen, J.Rudewig, and V.Stefanuk AMICA toolkit Architectural Design, Issue 2. Topical Report D43, Laboratory for Man-Machine Systems, University Kassel, BRITE/EURAM Project 6126, September 1994.

[10] Stefanuk V.L. Some aspects of Expert System theory//Soviet J. Comp. Sci. (Tech. Kibernet.),Moscow,1987. - N.2. - pp. 85-91.