X-AiD: A SHELL FOR BUDLDINO HIGHLY INTERACTIVE AND ADAPTIVE USER INTERFACES

Hans-Werner Hein, Gen M. Kellcrmann, Christoph G. Thomas

GMD Schloss Birlinghoven
D-5205 Sankt Augustin, West Germany
hein@gmdzi.uucp

ABSTRACT

X-AiD is an intelligent shell to design and operate user interfaces. Its surface offers an integrated and extendable structure of icons, windows, menus, natural text, universal operations, and special service functions. X-AiD consists of a set of specialized knowledge interpreters, implemented as asynchronous processes exchanging messages.

X-AiD operates based on declarative knowledge. It embodies common sense about "working with a computer" in general, expert knowledge about the supported applications behind it, and collected knowledge about each of its users. Knowledge is described using the representation language HAL. HAL enables comfortable declaration of object schemes with multiple inheritances and extensive default-handling. All HAL object schemes may contain specialized rulcsets related to human-computer communication topics (eg. syntax, semantics, display).

The system is prepared to work in a "learning by being used"-mode where it memorizes protocols about all occuring actions, including undo/redo operations. Later on it analyses those protocols extracting new object schemes, fluctuation of defaults, new semantics rules, and frequent plans-of-action which the specific user mainly followed. This learned knowledge is applied vice versa to aid this user eg. by preparing for him situation-dependent menus and object instances down the mainstream of his work or explaining to him his dialog state and how he got there.

L Introduction

Human-computer interfaces are commonly considered to play a very important role in future computer systems. Many statements concerning this are made. We at GMD felt already 1983 that besides the classical field of Natural Language Processing AI can contribute much to the development of smarter human-computer interfaces. So with the beginning of 1984 the research group Man-Machine Communication at the GMD Institute for Applied Information Technology started developing a system called AiD [1]. The following primary system characteristics then had been defined:

* AiD is a computer interface which is special to each user but similar for any application.
* AiD is a knowledge-based system which learns "by being used" and is able to act on its own initiative.
* AiD is an application independent assistant for users of any stage of computer literacy.

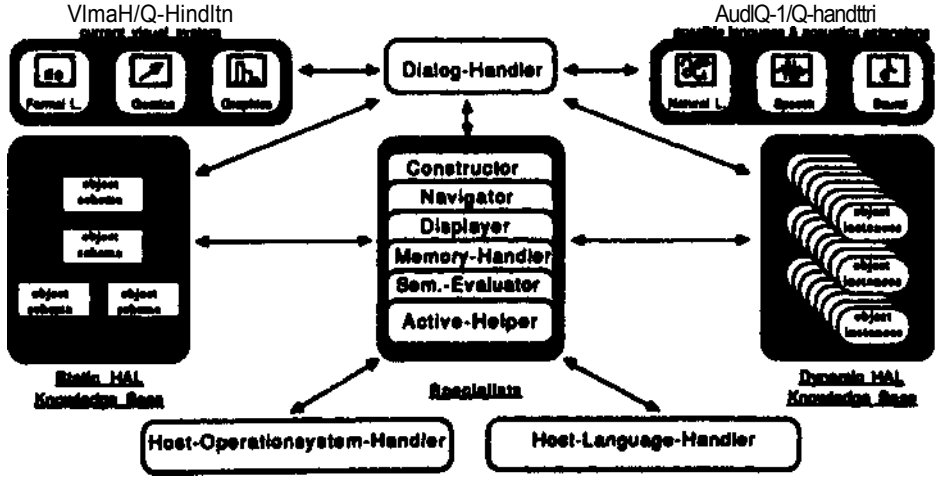
The resulting system, now X-AiD, has been implemented on Symbolics 36xx machines using CommonLisp and Flavors for the functional kernel. All the Knowledge Bases and other parts of the X-AiD system are declared using the knowledge representation language HAL. X-AiD provides users with a rich set of facilities including rapid prototyping with HAL, structure-oriented editing of object instances, different UNDO/REDO strategies, following predefined plans-of-action, holoptrasted displays, active help, and different types of learning.

2. X-AiDSyiiem Overview

The main functional X-AiD components are a collection of asynchronous knowledge interpreters fSpecialiati) and some additional programs (Sfiyjfifil). The main declarative X-AiD components are the Sliik and Dynamic Knowledge B«CJ described in the language HAL.

The Specialists are providing the knowledge-based support of a user's dialog with his collection of interactively manipulateable object instances in the Dynamic Knowledge Bases. We call the set of all user-accessible operations of the Specialists the Univefttli (or UnIver-I Opgratinml of X-AiD. Specialists in the X-AiD repertoire, generally applicable across a wide range of X-AiD applications, are handlers for:

* dialog with the user,
* input/output according different media,
* memory of object schemes and instances,
* displaying independent of the medium,
* navigating within the logical structure of object instances,
* constructing operations inside object instances,
* semantical evaluations across larger parts of the knowledge bases,
* active helping.



Wt^nl-Y-AiDAichi

Fig. 1 shows the modular design of the X-AiD system with the Dialog Handler at the center serving a monitor role in initiating and coordinating communication among the other Specialists. Not shown in this diagram are the collection of rule interpretation strategies accessible to the Dialog Handler and to each of th* Specialists in the system, together with the Knowledge Bases over the user and dialog history, each of which are (potentially) consulted at every interaction cycle with the user in order to interpret his command with respect to the current state and the diverse knowledge embedded in the computing environment.

It has been an important milestone in the X-AiD development when in 1986 we finished the implementation of the first versions of Constructor, Navigator, and Displayer. These three Specialists are forming an universal syntax-directed editor for instances of any possible HAL object scheme (for more about Specialists see [2]).

3. The Knowledge Representation Language HAL

The Knowledge Bases of X-AiD consist of objects declared by using the knowledge representation language HAL [3J and are organized in three dimensions:

* first they are divided into Static Knowledge Bases, containing object schemes and Dynamic Knowledge Bases, containing object instances,
* second they are modularized thematically according to a semantical classification of the objects,
* third they are grained by means of the object formalism itself, because in HAL all represented knowledge for any object strictly is contained inside this object.

Fig. 2 shows an example of a HAL object scheme ("Town") and the resulting default display *of* a 'Town" instance.

```
(OBJECT-SCHEME Town :PUBLIC

  ^DOCUMENTATION (purpose "IJCAI Town example"))
  (•.INHERITANCE X-AiD-basic-object)
  (rATTRIBUTES
    (postal-code :DISPLAY (:INSTANCE-OF Primitive-String)
                                                    : UNDEFINED)
    (Town-name :DISPLAY (:INSTANCE-OF Identifier)
                                                    UNDEFINED)
    (country     :DISPLAY (:P01NTER-TO Country)  UNDEFINED)
    (icon        .HIDDEN (:INSTANCE-OF Icon)
                 ((picture-file "aid-host: aid; graphics; icons; town.pic")
                 (width 79) (height 69))))

  OSYNTAX
    ((postal-code)          -> (:OR european-postal-code us-zip-code))
    ((european-postal-code)-> ((:AND {country-code"-"}
                                                    COPT 1 5 digit))))
    ((country-code)     -> (:OR " I " "GB" "F" "D"))
    ((US-zip-code)       -> ((:AND US-country-code (:OPT5 5 digit))))
    ((US-country-code)  -> (:OR "NY" "CA" "MA" "TX" "PA" "MD"   ))
    ((digit)               -> (:OR "0" " 1 " "2" "3" "4" "5" "6" "7" "8" "9")))

  (rSEMANTICS
    ((:AND (:IS country-name "Italy"))
          -> (:AND (<-town-name "Milano") (<- postal-code "1-20123"))
    ((:AND (:IS Town-name "Los Angeles"))
          -> (:AND (<-postal-code "1-20123") (<- country USA))
    ((:AND (:IS postal-code "D-7500"))
          -> (:AND (otown-name "Karlsruhe") (<- country W-Germany)))

  OHEURISTICS)
  (tDISPLAY)

END Town)
```

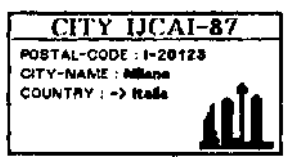


Figure 2: A HAL object scheme and instance

HAL is easily extendable with new features, i.e. the current slots are not fixed ultimately. For example a new slot :ALIASES may be useful in realizing a multi-language user interface, in which for each English object-name and attribute-name there is specified a Japanese/German/Arabic etc. alias.

Within X-AiD a very wide range of knowledge is represented in HAL:

* common sense about the business world, office environment and human communication (e.g. object schemes "Organization", "Document"),
* knowledge of prototype applications concerning spreadsheets (CalcAiD), electronic mail (MailAiD), papers (AuthorAiD) etc.,
* system internals, e.g. knowledge necessary for communicating with and supporting the user(dialog-management. user-profile, plan-of-action,...).

It is a long-term goal of the X-AiD-project to describe as much of the shell itself in terms of HAL object schemes and instances. This facilitates a portation of X-AiD to target computers, since only the procedural kernel of the system, written in the "development host language" CommonLisp, needs to be transformed into other programming languages ("target host languages").

Currently we are developing a declarative description of HAL in HAL, that yields a syntax directed editor for HAL schemes, when finished. An interesting consequence of "HAL-in-HAL" is the fact, that this consequently removes the difference between schemes and instances (at least in theory), because all HAL schemes become instances of the root scheme "HAL-in-HAL". With "HAL-in-HAL" we create a remarkable object scheme which is recursive in the sense that it is an instance of itself.

A main feature of HAL is the modularization of the rules first by inserting them into the object schemes and second by dividing them into different classes. A common property of all strategies and of all HAL rule interpreters is the independence from rule-ordering. Therefore it is not possible to hide an algorithm in a rule sequence.

The :SYNTAX rules take the form of productions of a context-free grammar with an attribute name as the start symbol. They restrict the feasible values of the attribute in question to the formal language specified by the relevant rules. When the user inserts or alters a value for this attribute, the typed input is parsed character-wise by an interactive parser-generator associated with the :SYNTAX slot of the HAL scheme.

The SEMANTICS rules permit the examination and comparison of attribute values on the left-hand side and the setting of them (perhaps with access to some other attribute values) on the right-hand side. There are different strategies available for interpreting semantics rules: DO ALL, DO ONE, WHILE ALL, WHILE ONE as in LOOPS [4]. The programming of side-effects in the rule's action part is enabled through hostlanguage calls, e.g. "(:LISP <arbitrary CommonLisp expression>)".

The ;HEURIgTICS rules are used for controlling the dynamic behaviour of instances. With heuristics rules the functioning of the universals can be adjusted to a particular task. While the X-AiD Universals are the default meiliojis of the object-based language HAL, the :HEURISTICS rules offer a controllable opportunity to include application-dependent special methods to the X-AiD Knowledge Bases.

The :DISPLAY rules fit for organizing the graphical and textual layout of the instances the user works with. This will be expanded systematically by acoustical and linguistical display rules. It is not necessary to specify some display rules - without them the Displayer Specialist produces a default layout that is sufficient in most cases.

All four rule sets are included in the inheritance mechanism, i.e. in HAL not only the attributes are inherited but also the rule sets, according to their classification. So the :SYNTAX rule sets in an inheritance line are joined separately to a new more specific :SYNTAX rule set, the :SEMANTICS rule sets are joined separately, and so forth. The important advantage of this thematic rule modularization is a very significant gain in effiency, since only the rules relevant to the current thematic context have to be interpreted when performing a X-AiD universal operation on a HAL instance. Therefore the rule interpretation remains efficient, even if the total number of rules in the Knowledge Bases becomes very large (i.e. millions).

5. Adaplivcness

Man-machine surfaces built with X-AiD may be viewed as intelligent personal aSSJSemS which should know many facts and rules about their users to be able to help them appropriately. You may consider X-AiD plus its Knowledge Bases about an user X to be an expert system for the needs and tastes of user X. Wherefrom should the knowledge come? There is no human expert to be asked about user X and X neither knows what he really wants nor ever has spare time to think about this topic. The only feasible way was to make X-AiD learn itself what it needs to know, starting from a default user profile. This user profile might be a description (made by knowledge engineers) of any existing unintelligent well-known interface, e.g. the Apple Macintosh's. X-AiD then first of all simply emulates such an interface.

Now the user may change things using the prototyping features and X-AiD will remember those changes. But besides this trivial "rote learning" the interface operates in a mode we named "learning by being used". Anything happening at the computer surface is protocolled scmamically using the plan-of-action scheme. This is very different to keystroke-level protocols which only contain purely syntactical information. The protocols are a primary source for the learning mechanisms of X-AiD. Evaluation by graph pattern matching and statistics is performed during longer periods of user inactivity. Based on the HAL knowledge representation language and the plan-of-action scheme the following will be learned 15]:

* new or variatod HAL object schemes,
* new or revised attribute defaults,
* semantics rule hypotheses,
* abstracted plans-of action.

6. Learning oj .object schemes

With the surface of X-AiD based on the language HAL it is possible to enable users to create object instances within the given limits of those object schemes already contained in the Knowledge Bases. If we now add the description of a totally universal object scheme (Fig. 3), users as well as the X-AiD system itself may create any possible HAL instance avoiding to declare some specific object scheme. Virtually the memory now contains instances of arbitrary kind without a joint scheme. Those orphaned instances can be handled by X-AiD like any others.

At any time the orphaned instances can be sampled from the memory and compared by pattern matching techniques. If it appears that some instances are structurally equivalent, this equivalence class can be described in a new machine-generated HAL scheme. The universal-scheme instances of that cluster are then to be transformed to instances of that newly learned and maximal specific HAL object scheme.

```
(OBJECT-SCHEMIE universal-scheme :PUBLIC
  (:INHERITANCEX-AiD-basie-objcct)
  (: ATTRIBUTES
    (inheritance .DISPLAY (:SET-OF universal-scheme) .UNDEFINED)
    (attributes  :D1SPLAY (:SET-OF universal-altributc)UNDEFINED)
    (syntax      .DISPLAY (:SET-OF universal-rule)    UNDEFINED)
    (semantics   .DISPLAY (:SET-OF universal-rule)    UNDEFINED)
    (heuristics  :D1SPLAY (:SET-OF universal-rule)    UNDEFINED)
    (display     :D1SPLAY (:SET-OF universal-rule)    UNDEFINED))
END universal-scheme)

(OBJECT-SCHEMIE universal-attribute INTERNAL
  (:INHERITANCEX-AiD-basic-object)
  (:ATTRIBUTKS
    (name    .DISPLAY Identifier                          UNDEFINED)
    (type    :DISPLAY (POSSIBLE-VALUES
                        UISPLAY :READ :H1DDEN) UISPLAY)
    (scheme  :DISPLAY universal-scheme                    .UNDEFINED)
    (default UISPLAY universal-scheme                     UEFAULT))
END universal-attribute)

(OBJECT-SCHEMIE universal-rule :INTERNAL
  (INHERITANCE X-AiD-basic-object)
  (•ATTRIBUTES    (name    :D1SPLAY Identifier UNDEFINED)
                  (left-side. . . . . .)

 etc.
END universal-rule)
```

Figure 3: HAL-declarod inventory to handle "orphaned instances"

7. Learning of semantics rules

Semantics rules typically define consistency between attribute values of an object instance. Those values may be parts, i.e. simple or complex instances inside, or pointers to other independent instances with non-hidden attributes. All these accessible pathnames may be used in the condition predicate of a semantics rule. If an inconsistency occurs, X-AiD knows three types of handling this.

One is the "lazy mode", where X-AiD just waits until its user finishes an expected series of value changing actions. The second mode is to automatically recover consistency by executing the rule's action part. This is the "expert mode", useful for diagnosis and simulation subtasks.

The third and most interesting type of inconsistency handling is using the whole rule as an explanation source in the "helper mode". X-AiD then asks its user if he wants to correct some of his last inconsistency-causing value changes, or if he insists in his changes. If the user declares in such a situation the new object instance state to be consistent, X-AiD has to modify or delete its now "user inconsistent" semantics rule.

This prerequisite behaviour enables a learning mode where X-AiD hypothesizes semantics rules about specific object schemes. It only requires a statistical significant number of instances. Then cluster analysis and classifier dimensioning methods from the Pattern Recognition field [6J are easily applicable because the therefore often questionable necessary set of "simple constituents" here always is algebraically at hand: It is the focussod object schemes complete HAS-A relation.

If a semantics rule, hypothesized statistically, stays "user consistent" for a while (i.e. it is used suiccssfully several times in "helper mode") then it may be used by X-AiD in "expert mode", too.

8. Conclusion

Based on the operational level of X-AiD in its current state, research and development now will concern mainly these goals:

* including acoustical and speech media in the user's communication surface,
* installing a language switch feature, where a user may switch X-AiD's language, fonts, and cultural habits just "pushing a button",
* integrating the now separate types of learning and some more types to a synergic adaptive system.

We like to thank all other colleagues and students of the AiD Team for their contributons to the project, in particular Elke Finkc, Uwe Horn, and Juorgen Krcuziger.

REFERENCES

111 Hein, Hans-Werner, Smith, Scott R., Thomas, Christoph G.:
AiD improves Dialogs - A Better Approach to the Design of Man-Machine Dialogs through Knowledge-Based Techniques.
European Symbolics Users Newsletter. Vol.1, No.2, Nov. 1984.

[2J Thomas, Christoph G., Kellermann, Gort M., Hein, Hans-Werner:
X-AiD: An Adaptive and Knowledge-Based Human-Computer Interface.
Submitted to INTERACT-87 to be held in Stuttgart, West Germany.

13) Thomas, C G., Hein, Hans-Werner:
The HAL Manual.
GMD Technical Report, Sankt Augustin, to appear 1987

[4] Bobrow, Daniel G., Stefik, Mark:
The LOOPS Manual.
Palo Alto, Califomia, XEROX PARC, 1983.

[5] Michalski, R.S., Carbonell, J. G., Mitchell, T. M. (Eds.):
Machine Learning: An Artificial Intelligence Approach I, II
Morgan Kaufmann, Los Altos, Califomia, 1983 &1986.

[6] Fu, K. S.:
Sequential methods in Pattern Recognition and Machine Learning.
Academic Press, New York, 1968.