The AREDIT Interlisp bug database system

author: Michael Sannella
files: `{eris}<lispusers>AREDIT.DCOM`
doc: `{eris}<lispusers>AREDIT.TXT`
uses: All Tedit files

The file `AREDIT.DCOM` contains a number of tools useful for examining, editing, and submitting ARs ("Action Requests") related to the Interlisp-D system. These tools are loosely based on the "Adobe" tools in the Tajo environment. The Interlisp-D support group uses this system to keep track of the state of outstanding bug reports. There are currently over 2000 ARs in the database.

These tools can be used from any machine running Interlisp-D which can establish a leaf connection to the `PHYLUM` file server, where the database files are currently stored.

After loading `AREDIT.DCOM`, the user can create two types of windows: AR edit forms and AR Query forms.

The AR Edit Form

An AR edit form is used to examine, edit, and submit ARs. To create an AR Edit Form, evaluate (AR.FORM). Interlisp will prompt you to specify a region for the form window -- the best size to give it is one about half the width of the screen and at least half the height of the screen. The form window which will appear contains three subwindows: (1) On the top is the message subwindow, where prompts and status messages are printed; (2) in the middle is the command subwindow, a menu of commands for editing / submitting ARs; (3) on the bottom is the form subwindow, where the information in an AR is displayed.

The command subwindow contains the following commands:

`New` -- Buttoning this word clears the fields of the AR in the form subwindow. Some fields (**`Source, Submitter, Status`**) are initialized to appropriate values for a new AR.

`Get` -- Buttoning this retrieves the AR whose number follows "**`Number:`**" in the command subwindow.

`Put` -- Buttoning this will either store an edited AR, or submit a new AR. Which one (submit new or store old) depends on whether the last operation was "**`New`**" or "**`Get`**". If the current AR displayed was retrieved with "**`Get`**", then "**`Put`**" will store it as the old AR. If this AR was built up from scratch after buttoning "**`New`**", then "**`Get`**" will submit is as a new AR. The title of the form subwindow gives an indication of what state the form is in: if it says "New Bug Report", then "**`Put`**" will submit it. If it says "Editing AR xxx", then "**`Put`**" will store it. [There are plans to improve this interface]

`Number:` -- This is a text field just like text fields in the form window (see below) used to specify the number used by "**`Get`**". Buttoning the word "**`Number:`**" will pending-delete-select the value of the field, so you can delete it and insert a new number. If the character carriage-return is typed, then a "**`Get`**" is automatically done on the value of this field. This is faster than typing a number, and buttoning "**`Get`**".

The form subwindow contains a large number of fields. The meaning of these fields is described in `{phylum}<lispars>LispARFields.press` . The value of these fields can be edited as follows:

"Enumerated fields" can only contain certain values. These are indicated in the form subwindow by field names followed by curly-braces "`{}`". To change the value of one of these fields, button the field name; a menu of permissable values will appear; select a value; it will be inserted between the braces.

[note: Some of the enumerated field values are dependent on other fields. For example, the values of "Subsystem:" depend on the value of "System:" -- if the "System:" value is changed, the "Subsystem:" value is automatically set to NIL. The fields with this relationship are System:/Subsystem: and Machine:/Disk:.]

"Text fields" can contain arbitrary text. These fields do not have braces after the field name. The text can be edited using normal Tedit editing. Buttoning the field name will pending-delete-select the entire field value, which allows the whole field to be easily deleted.

[note: Currently, stored ARs only contain straight text. Any tedit formatting information put into an AR will be lost when the AR is stored. Image objects (like bitmaps) are also not stored.]

[note: A few of the text fields, like "Number:", are read only --- they cannot be edited by the user.]

[note: in older versions of AREDIT, some text fields (such as "Attn:" could only contain a certain number of characters. The user could type as many characters as he wanted, but an error would occur when the "Put" command was executed. This has now been changed --- any text field can contain an arbitrary number of characters.]

The AR Query Form

An AR Query Form is used to search the AR database for all ARs with particular characteristics. One can search for all ARs with a given name in the "**`Attn:`**" field, all ARs which have `Status: = Open`, etc. These ARs can be sorted, and a summary of the selected ARs can be printed into a file.

To create an AR query form, evaluate `(AR.QFORM.CREATE)`. Interlisp will prompt for a region (the default size is ok), and create a window with three subwindows: (1) on top, a message subwindow, for printing prompts and messages; (2) in the middle, a browser subwindow, used for displaying the ARs seleced by a query; (3) on the bottom, the AR query command subwindow, containing a number of commands and fields.

The Ar Query command subwindow contains the following fields/ commands:

`Query List:` -- This field is used to specify which ARs the "**`Query`**" command will search for. This field should be filled with an AR query spec, which has one of the following forms:

`(<field> HAS <val>)` searches for all ARs whose text field <field> contains <val>. <string> may either be an Interlisp string or an atom. The search is case-independent: foo matches Foo matches FOO.

`(<field> IS <val>)` searches for all ARs whose enumerated field <field> has the value <val>.

`(AND <spec1> <spec2> ... <specN>)` returns all ARs satisfying ALL of the given specs.

`(OR <spec1> <spec2> ... <specN>)` returns all ARs satisfying ANY of the given specs.

[note: an implicit `(AND)` is wrapped around the value of the "Query List:" field, so just giving a number of specs will AND them together.]

Not every AR field can be searched for in the same way: some can only be searched with HAS, some can only be searched with IS, and some (like the Description: field) cannot be searched at all. To find out the possible query specs, button the words "**`Query List:`**" --- this will put up a menu of all of the permitted searching options. Some of these menu items have submenues. When one of the options is selected, it is added at the end of the value of this field.

Examples:

`Query List:` `(Subject: HAS foo)`
Searches for all ARs whose subject contains the string "foo".

`Query List:` `(Status: IS Open) (Attn: HAS sannella)`
Searches for all open ARs which have "sannella" in the Attn: field.

`Query List:` `(OR (Status: IS Declined) (Status: IS Superceded) )`
Searches for all ARs with Status: either Declined or Superceded.

`Sort List:` -- This field determines how the Query-ed ARs will be sorted. Currently, ARs can only be sorted by the values of enumerated fields. Buttoning the words "**`Sort List:`**" will put up a menu of the permitted field names -- selecting one will add it to the value of this field.

Example:

`Sort List:` `Status: Priority:`
This will sort first by the `Status:` field, and then by the `Priority:` field.

[note: After sorting by all given fields, if two ARs are the same, they are sorted by AR number. Therefore, if this field is left blank, the queried ARs will be in numerical order]

`Query` -- Buttoning this command will initiate the query specified by the "**`Query List:`**" field, and sort it according to the value of the "**`Sort List:`**" field. While the query is in progress, the AR query command subwindow is greyed-out. When the query is completed, the numbers, subjects, etc of the ARs which have been found are displayed in the AR query browser subwindow. This window can be scrolled both vertically and horizontally.

`Print File:` -- This field can be filled in with a file name, which is used to specify the file that the **`Print`** command should store a report. If left blank, a window will pop up on the screen, and the information will be displayed there.

`Print` -- This prints a detailed summary of all of the ARs from the last **`Query`** into the file given in the **`Print File:`** field.

To generate and print a summary of a selected group of ARs, fill in the **`Qury List:`** and **`Sort List:`** fields, select **`Query`** and wait for the query to complete, fill in the **`Print File:`** field, and select **`Print`**. The summery is rather wide -- it may be a good idea to use the `LANDPRESS` package to printit out sideways on a printer.

The AR query browser window:

This window shows a short summary (one line each) of the ARs that have been queried. Left-buttoning one of the AR lines will call `AR.SHOW` on that AR, to display it. Middle-buttoning an AR line will "**`Get`**" that AR into a specified AR edit form window.

Background menu commands:

When `AREDIT` is loaded, the item "`AREDIT`" is added to the background menu, with a number of subitems. These are interpreted as follows:

`AREDIT` -- Creates a new AR edit form, initially cleared.

`New AR Form` -- Same as `AREDIT`

`Load AR Form` -- prompts the user for a number, and creates a new AR edit form, with the specified AR number loaded initially.

`AR.SHOW` -- prompts the user for a number, and calls `AR.SHOW`, an old version of `AREDIT` which quickly displays the contents of a given AR. It prompts for a window region the first time it is used -- thereafter it uses the same window.

`AR Query Form` -- Creates a new AR query form.

Auxiliary AR edit form commands.

Pressing the left mouse button in the title bar of the AR Edit form command subwindow will bring up a menu of less-used commands:

`Clear` -- Clears ALL the fields of the current AR. Similar to `New`, except that non of the fields like `Source:`, etc., are filled in. This is useful when you are submitting an AR for someone else.

`New`
`Get`
`Put` -- the same as the Commands in the edit command subwindow.

`Put&Get` -- prompts the user for a number, `Put`s the current AR, and `Get`s the given numbered AR. Useful when scanning through a number of ARs.

`GetFromFile` -- Prompts the user for a file name, and loads the information from that file into the AR edit form subwindow. If this file is not in the right format, an error will be generated.

`PutToFile` -- Prompts the user for a file name, and stores the information from the AR into that file.

Locally caching the AR index.

All AR query operations use the information in the "AR Index" file. This file, which is updated every few days, is stored as `{phylum}<lispars>AR.INDEX`. To speed up Query operations, this file can be copied to a local file server, or the local hard disk. Warning: this file is currently ~700 IFS pages long, and it will undoubtedly get larger. Also, it is the responsibility of the user to make sure that they update their local version of `AR.INDEX` when the master copy is updated.

To use a local version of `AR.INDEX`, give the file name as an argument to `AR.QFORM.CREATE`:

```
(AR.QFORM.CREATE '{DSK}AR.INDEX).
```

Global Variables that control AREDIT

The following are global variables that can be set to alter the operation of AREDIT. These are the only global variables that it is safe to change.

`AR.ENTRY.LIST.WINDOW.FIELDS` -- Controls which fields are displayed in the AR query browser window, along with the widths of the fields.

`AR.ENTRY.LIST.PRINT.FIELDS` -- Controls the fields displayed by the Print command of the AR query form.

`AR.ENTRY.LIST.PRINT.MULTILINE.FLAG` -- if non-NIL, the Print command of the AR query form will print all of the characters in each field, using multiple lines for those field values bigger than the field width allowed. If NIL, each AR will use only one line, truncating any field values that are too big.