The following notes explain my (woz) idea of what the edit-interface should be responsible for, and what constitutes and "editor".

An editor is a symbol who's function takes the args (structure props options) and starts an interactive editor on the structure. PROPS is a property list, and OPTIONS is a list of keywords, both affecting the behavior of the editor. The property :completion-fn specifies the function to be called when the user completes the edit. The completion-fn will be called with the arguments (structure props options changed?), where structure is the edited structure (even on abort, so that the edit interface could implement "undo abort"), props and options are as specified on the call to the editor and will be used to figure out what to do with this completion, and changed? is NIL (no changes made), T (changes made), or :ABORT (user wants to abort changes). It is then up to the completion function to do the right thing with the result of the edit.

The goal is that the editor doesn't know anything about who started it, where the structure came from, or what to do with it when it's done. And the edit-interface doesn't know anything about the editor's data structures.

In the case of open edit sessions (open or shrunk):
If the editor is told to start an edit, the editor must look for one already existing that matches (this can't be the responsibility of the edit interface, because it doesn't know about existence of open edits). the editor should restart the existing edit, processing any new props or options appropriately.

The markaschanged issue:
Since the editor knows it may have open edits, it needs to provide a hook for when the world gets changed underneath the editor. in this world this means markaschanged. in this case the editor should try to restart itself with the new structure. in other words, sedit::markaschangedfn should call edit-definition to start a new edit. the editor will then notice it has a matching edit open and restart itself with the new info.
Thismodel is complicated by the fact that markaschanged gets called as a result of completion. Presently *ingore-changes-on-completion* controls the behavior in this case. Ideally, the editor would just say "i know it got changed, i just changed it!", and it would ignore the call.

The new version of SEdit (1/25/91) is very close to this definition, with the following exceptions, which can be fixed upon implementation of this edit inteface design:
- the completion-fn is called with (context structure changed?) since all of SEDITE's completion-fns expect these args. this should be fixed in handle-completion.
- in the abort case, undo is run until there are no more changes to undo, since sedit is sometimes handed structures "in place", destructive edits need to be undone, and thus the completion-fn never gets to see the edits. this can be fixed in the function complete.
-

To make this editor active, call (il:editmode editor-name), where editor-name is the symbol defined above. The function xcl::edit will then start the active editor.

xcl::edit-expression provides an example of starting the editor on an unnamed structure, where eqness is expected upon completion.

xcl::edit-definition provides a replacement for il:editdef, if il:getdef and il:putdef work correctly on all types.

:dontwait
the editor (sedit) should not wait. don't wait is part of the editinterface features. if its going to wait, the editinterface should create the event and bind it in the completionfn to the notified by the completionfn.

need published edit-expression command. same as ed, but takes and expression, waits for once-only completion, and returns the expression. eg for FIX

sedit should provide a published interface to creating gaps, for use by make-prototype-defn.

edit interface may need a way to ask editor if it has an open edit for name/type

deal with editdates

an editor used to install itelf by replacing EDITL and EDITE, then editmode was created to look for Definition-for-EDITE property on the symbol returned by editmode. now an editor is defined as a function which takes (expr props options) and implements at least the props :completion-fn

and :root-changed-fn. EDITMODE then returns the name of this function so the current editor can be applied.