From: Trygve Reenskaug

Date: 10 December 1979

MODELS - VIEWS - CONTROLLERS

MODELS

Models represent knowledge. A model could be a single object (rather uninteresting), or it could be some structure of objects. ~~The proposed implementation supports knowledge represented in something resembling *semantic nets* (If I understand Laura correctly)~~

There should be a one-to-one correspondence between the model and its parts on the one hand, and the represented world as perceived by the owner of the model on the other hand. The nodes of a model should therefore represent an identifiable part of the problem.

The nodes of a model should all be on the same problem level, it is confusing and considered bad form to mix problem-oriented nodes (e.g. calendar appointments) with implementation details (e.g. paragraphs).

VIEWS

A view is a (visual) representation of its model. It would ordinarily highlight certain attributes of the model and suppress others. It is thus acting as a *presentation filter*.

A view is attached to its model (or model part) and gets the data necessary for the presentation from the model by asking questions. It may also update the model by sending appropriate messages. All these questions and messages have to be in the terminology of the model, the view will therefore have to know the semantics of the attributes of the model it represents. (It may, for example, ask for the model's identifier and expect an instance of Text, it may not assume that the model is of class Text.)

CONTROLLERS

A controller is the link between a user and the system. It provides the user with input by arranging for relevant views to present themselves in appropriate places on the screen. It provides means for user output by presenting the user with menus or other means of giving commands and data. The controller receives such user output, translates it into the appropriate messages and pass these messages on .to one or more of the views.

A controller should never supplement the views, it should for example never connect the views of nodes by drawing arrows between them.

Conversely, a view should never know about user input, such as mouse operations and keystrokes. It should always be possible to write a method in a controller that sends messages to views which exactly reproduce any sequence of user commands.

EDITORS

A controller is connected to all its views, they are called the parts of the controller. Some views provide a special controller, an *editor*, that permits the user to modify the information that is presented by the view. Such editors may be spliced into the path between the controller and its view, and will act as an extension of the controller. Once the editing process is completed, the editor is removed from the path and discarded.

Note that an editor communicates with the user through the metaphors of the connected view, the editor is therefore closely associated with the view. A controller will get hold of an editor by asking the view for it - there is no other appropriate source.