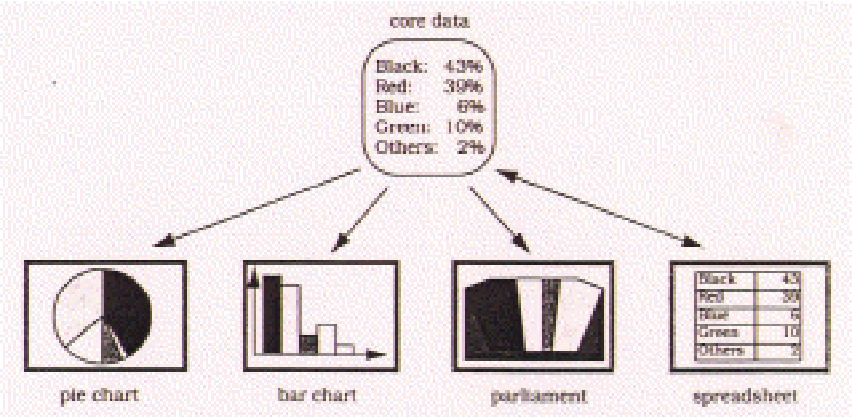
A Problem in User-Interface Design



Context

You are building an interactive application.

Problem

Programs that mix **functionality** with **interface** make it impossible to change one without changing the other.

Forces

This problem is really many problems rolled into one:

- Interfaces change more frequently than data models.
- Users request customized features or new interfaces altogether.
- Different users may have radically different needs.

If the data model and the user interface are interdependent, all of these problems become quite difficult to solve. A change to the interface will affect the data model implementation. A change in the data model will affect many or all of the interfaces.

A Problem in User-Interface Design

These are some of the forces at play:

- The same information is presented differently in different windows.
- The displays and system behavior must reflect data manipulations immediately.
- Changes to the user interface should be as simple as possible—even at run-time.
- Supporting different “look and feel” standards should not affect the core of the application (the data model).
- Porting the user interface to new platforms shouldn’t, either.

What we don’t want:

- To have to modify the application part of the program in order to change the user interface.
- To allow the user interface to have direct, unecapsulated access to the application data.

What we do want:

- To be able to change the application part of the program and leave the user interface alone.
- To be able to change the user interface and leave the application part of the program alone.
- To be able to have multiple interfaces on the same data—at the same time—without the different interfaces needing to know about each other.

A Common Solution to the Problem

Create a different kind of object for each role in the program:

- The *model* maintains the application's data and encapsulates its functionality. The model is independent of how the data is displayed to the user and how the user provides input to the system.

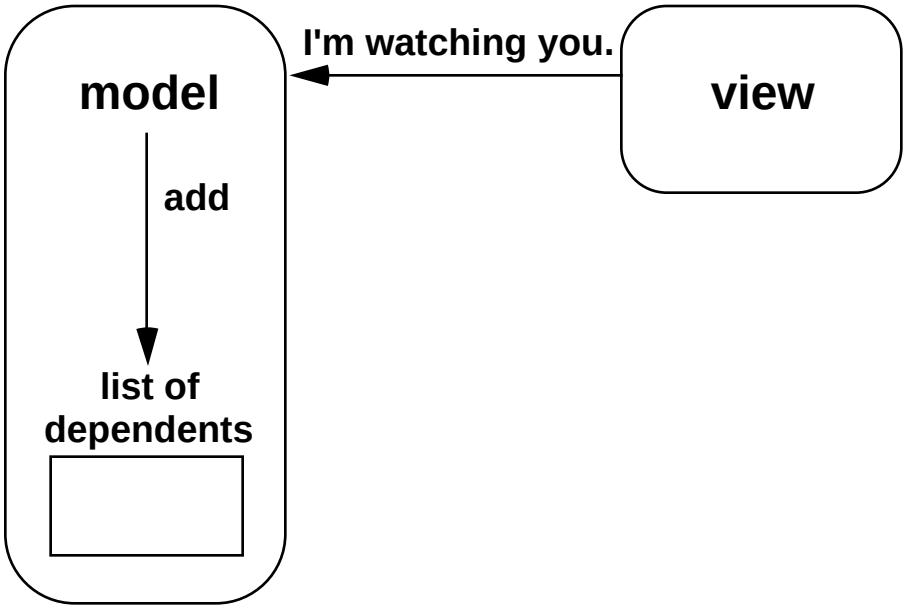
- The *view* displays data to the user. The view requests data from the model. There can be any number of views open on the same model at once.

- Each view has an associated *controller* that watches for input from the user. A controller receives user input, determines what the user is requesting, and then sends a request to the view.

If there are multiple views open on the same data model and the data change in some way (say, through user input), then all of the views should be updated immediately to reflect the change. In order for this to happen, the model must notify all views whenever its data changes. The view can then request any relevant data from the model and update its display.

The Change Mechanism

A view is created on a specific model.



Whenever the model changes one of its data values, it announces the change to all of its dependents. If the view cares about the change, then it asks for the data and and updates its display.