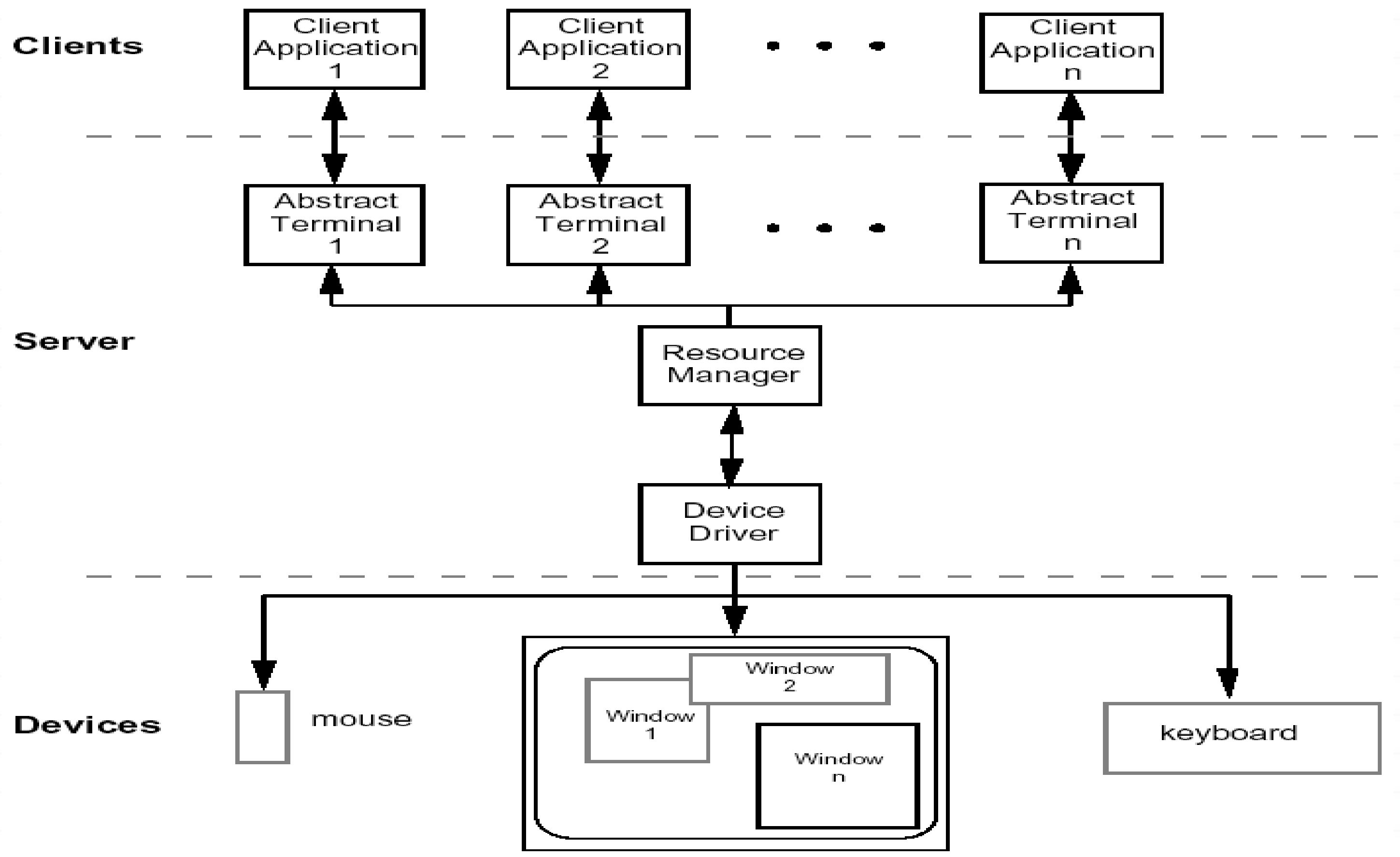
Chapter 8

Implementation Support

Elements of windowing systems

- Windowing System:
  - windowing system is a system for sharing a computer's graphical display resources/ GUI among multiple applications at the same time.
- Windowing Systems are:
  - Device/Hardware Independent
  - Support Multi tasking
  - Management of independent but simultaneously active applications
  - A windowing system will have a fixed generic language which is called **Imaging Model**.
- Only one program device driver needs to be written for a particular hardware device

Windowing System Architecture



The client-server architecture

Toolkits

- A toolkit provides the programmer with a set of ready-made interaction objects – alternatively called interaction techniques, gadgets
- These objects have a predefined behavior as that described for the button:

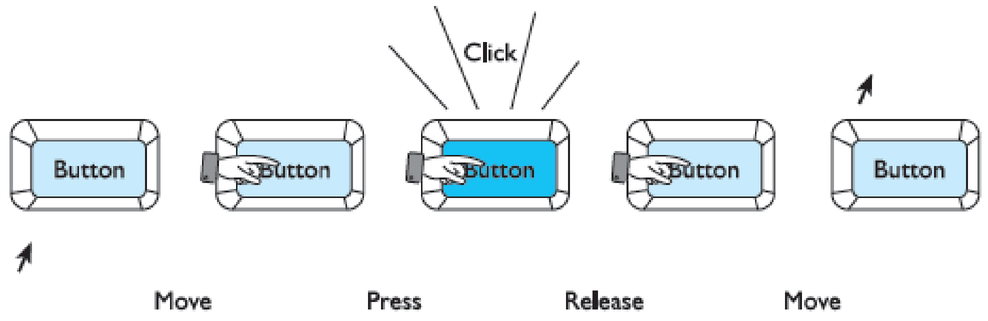


Figure 8.8 Example of behavior of a button interaction object

  - The Java toolkit for developing windowed applications is called the Abstract Windowing Toolkit, AWT

Toolkits (ctd)

- Toolkits provide Consistency and Generalizability for an interactive system.
  - One of the advantages of programming with toolkits is that they can enforce consistency in both input form and output form by providing similar behavior to a collection of widgets
  - This consistency of behavior for interaction objects is referred to as the look and feel of the toolkit

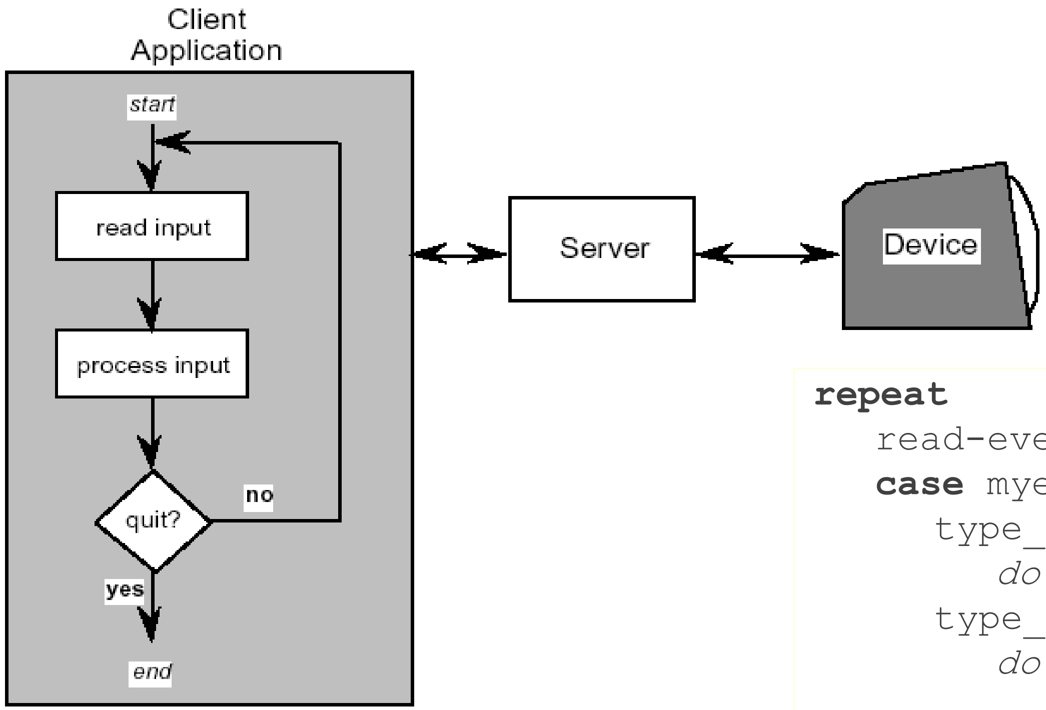
Toolkits

- To provide flexibility, the interaction objects can be modified
  - These objects are modified by --------------

- *Instantiation?*
- *Inheritance*
- *multiple inheritance?*
- *instance attributes?*

Programming the application - 1

read-evaluation loop



```
repeat
   read-event(myevent)
   case myevent.type
      type_1:
         do type_1 processing
      type_2:
         do type_2 processing
      ...
      type_n:
         do type_n processing
   end case
end repeat
```

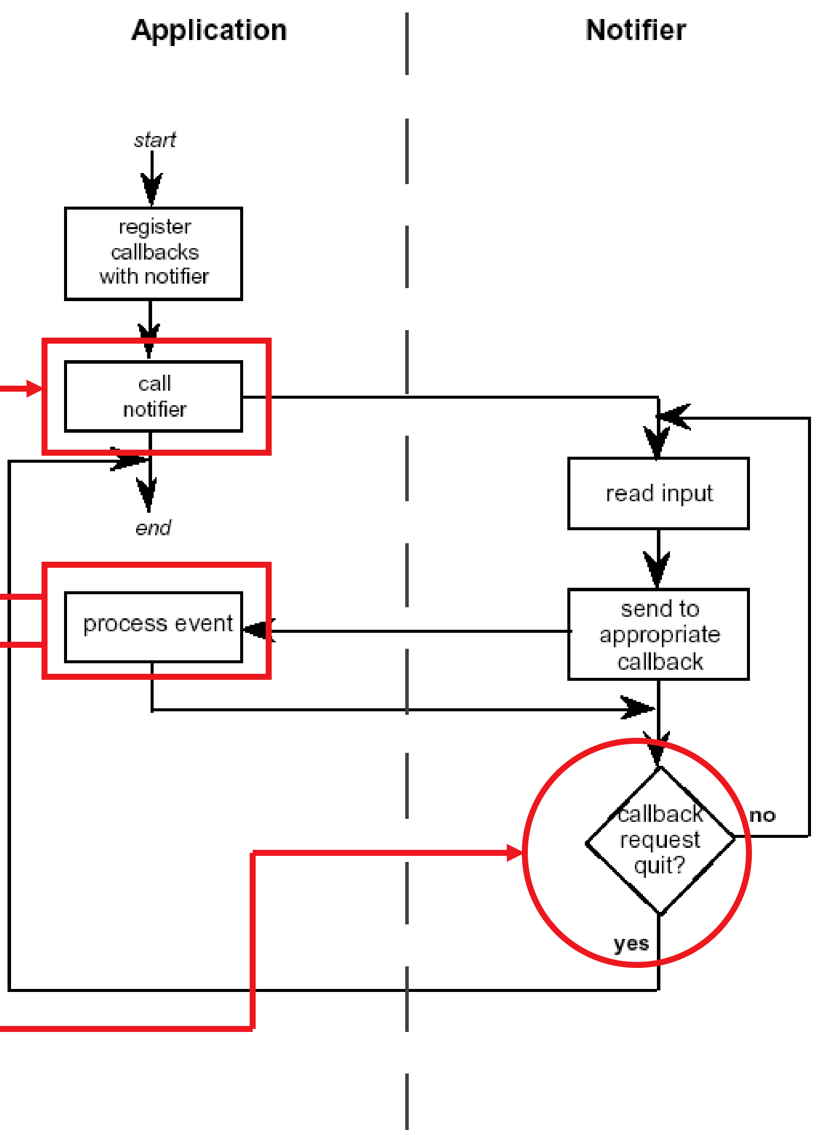
Programming the application - II

notificatio

```
void main(String[] args) {
   Menu menu = new Menu();
   menu.setOption("Save");
   menu.setOption("Quit");
   menu.setAction("Save",mySave)
   menu.setAction("Quit",myQuit)
      ...
}

int mySave(Event e) {
   // save the current file
}

int myQuit(Event e) {
   // close down
}
```



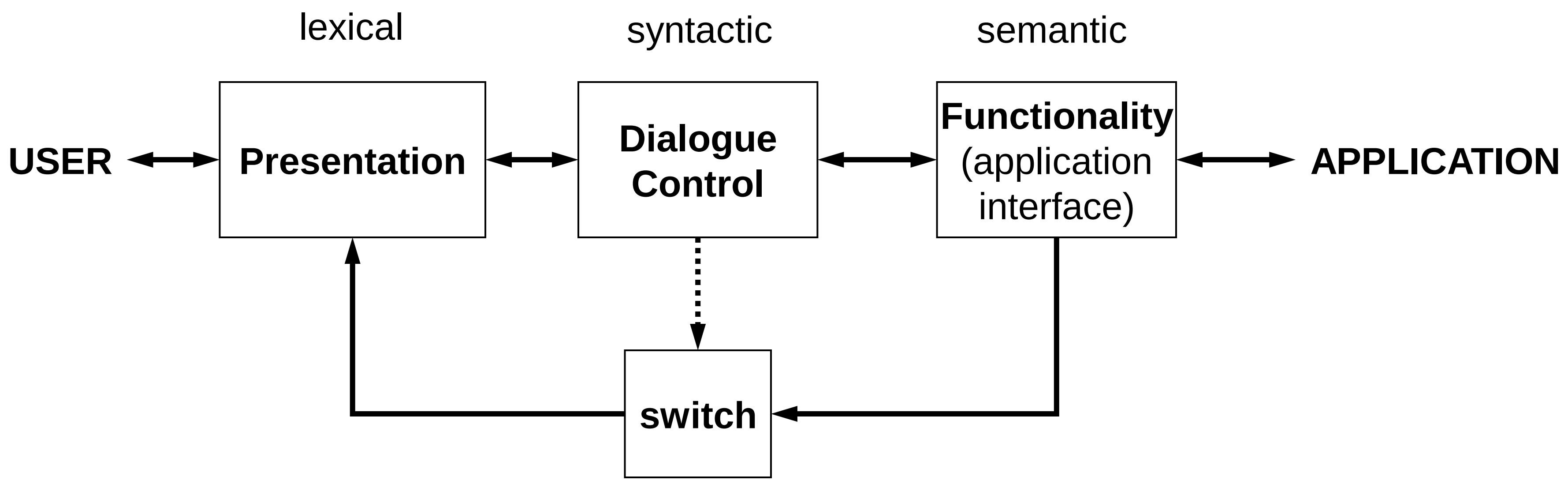
User Interface Management Systems (UIMS)

- The set of programming and design techniques which provide more development support for interactive system design beyond the toolkits.
  - Examples of UIMS are Serpent & Picasso
- The UIMS should support:
  - **Conceptual Architecture**:
    - for the structure of an interactive system which concentrates on a separation between application semantics/logics and presentation;
  - **Techniques:**
    - for implementing a separated application and presentation and preserving the intended connection between them;
  - **Support techniques:**
    - For managing, implementing and evaluating a run-time interactive environment

UIMS: Conceptual Architecture

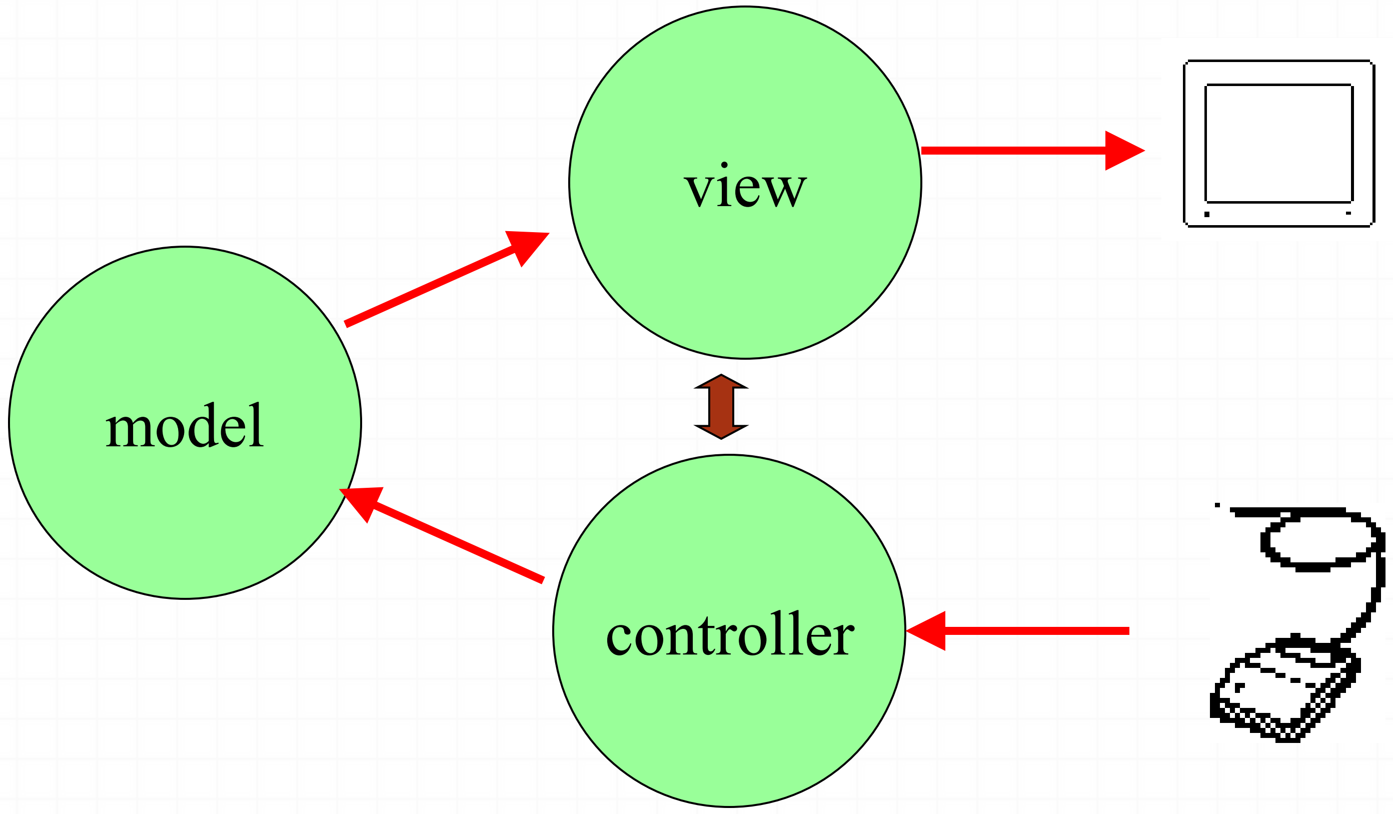
- Separation between application semantics/logic and presentation improving:
  - **Portability**
    - runs on different systems and provides device independent interface
  - **Reusability**
    - Reusability of components reduces development costs
  - **Multiple interfaces**
    - Supports development of multiple interface to access same functionality
  - **Customizability**
    - by designer and user without altering core application

UIMS: Conceptual Architecture



Seeheim Model

UIMS: Conceptual Architecture



MVC: model - view - controller

UIMS: Conceptual Architecture

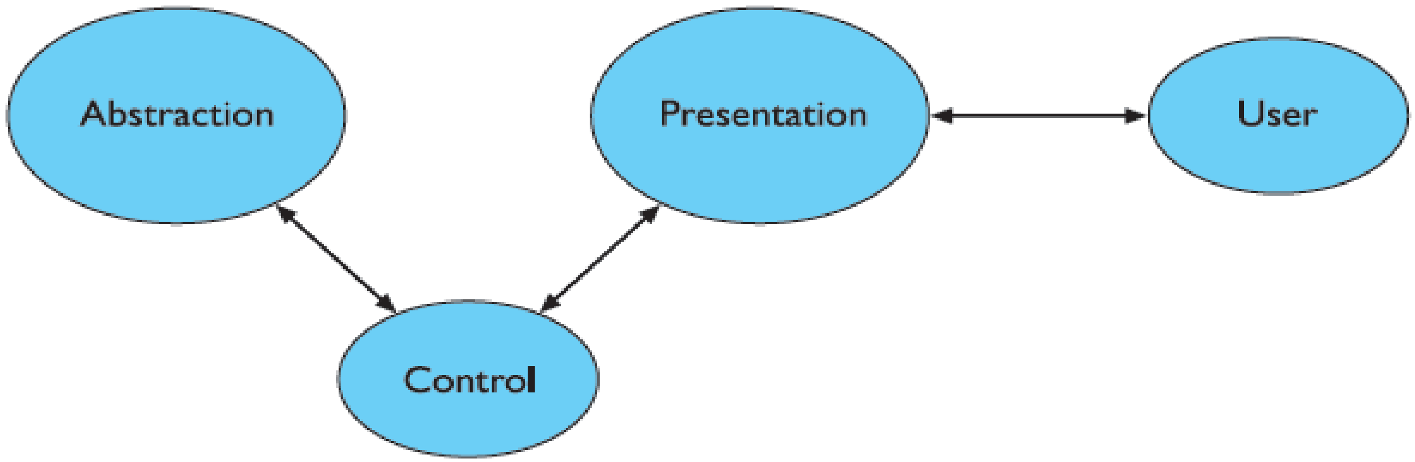


Figure 8.12 The presentation–abstraction–control model of Coutaz

Multi-Agent Architecture

PAC

presentation - abstraction - control

MVC & PAC Differences/Issues

PAC

- Groups input and output together.
- Secondly, PAC provides an explicit component whose duty it is to see that abstraction and presentation are kept consistent with each other.
- Not linked to any programming environment, though it is certainly helpful to an object-oriented approach.
  - It is probably because of this last difference that PAC could so easily isolate the control component; PAC is more of a conceptual architecture.

MVC

- whereas MVC separates them
- MVC does not assign this important task to any one component, leaving it to the programmer/designer to determine where that chore resides
- Cannot so easily isolate the control component

Implementation Considerations

- Dialog Modeling Techniques in UIMS:
  - Menu Networks:
  - Grammar notations
  - State transition diagrams
  - Event Languages
  - Declarative languages
  - Constraints
  - Graphical specification