CTT534 – Thiết Kế Giao Diện
HK II 2013 – 2014

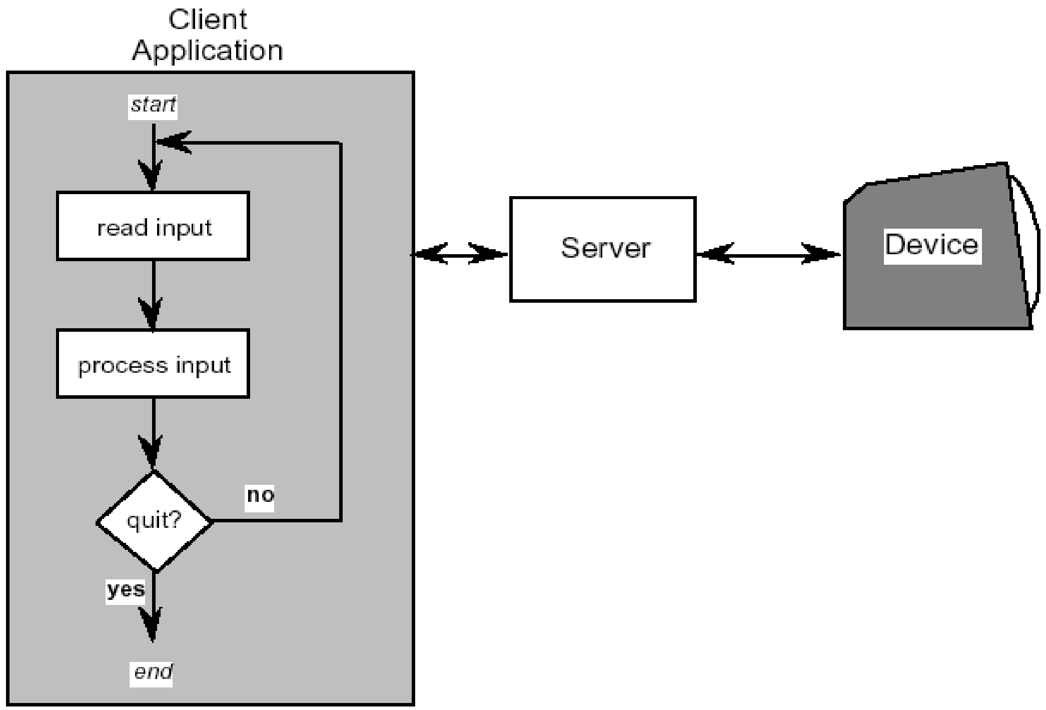
UI Architectures and Development Tools

Some slides adapted from materials of MIT CS Course 6.813/6.831

Outline

- Architectures and design patterns for UIs
- UI development tools

Read-evaluation architecture



Flowchart

```
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
```

Pseudo-code

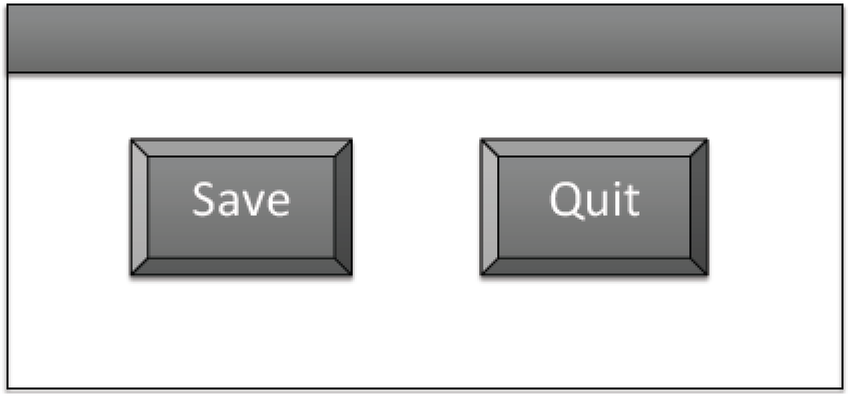
- The application has complete control over the event processing
- Programmer must execute this control over every event that the client will receive
  - very cumbersome task

Separating UI from application

- One aim of a UI architecture is to **separate the interface from the application**
- Provides
  - Portability
  - Reuse
  - Multiple Interfaces
  - Customization
  - Maintainability
  - Cost savings

Notification-based architecture

- Example
  - A simple interface for a “Save” and “Quit” operation

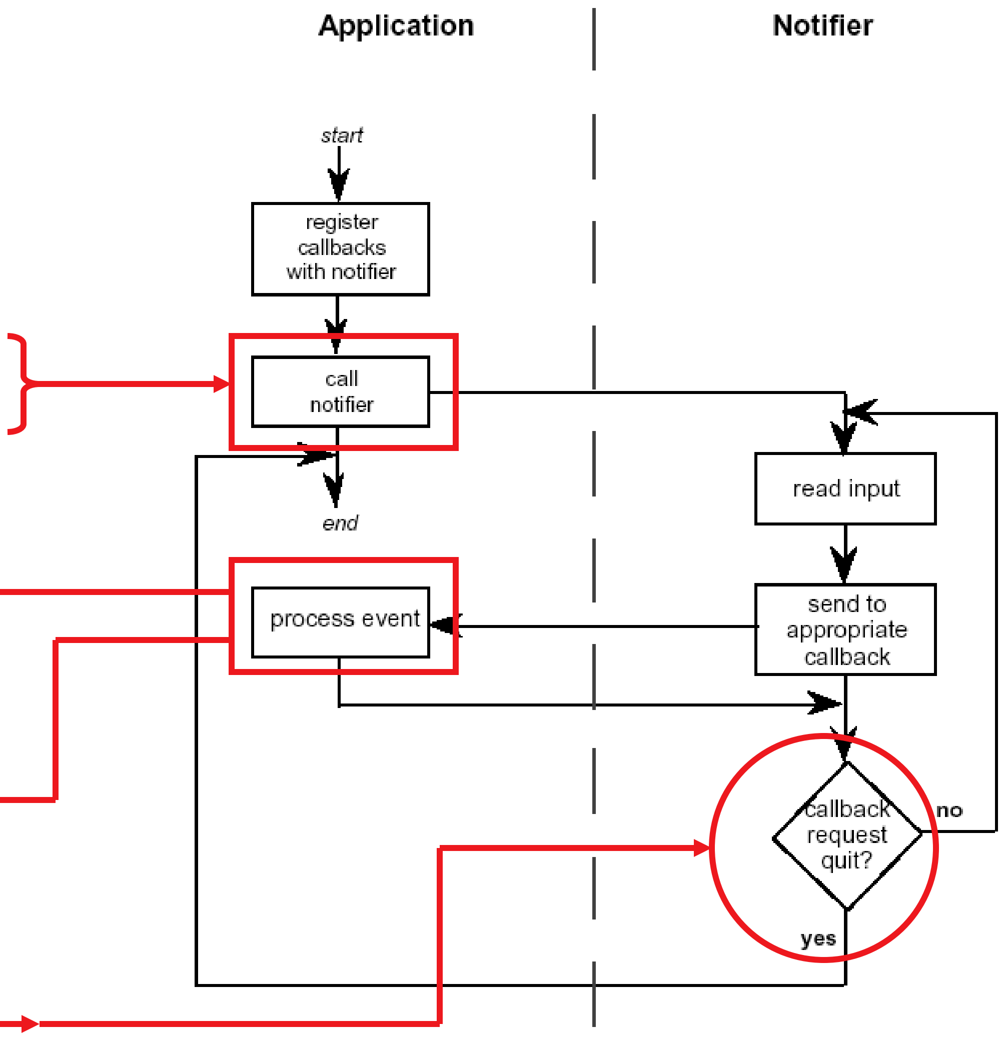


Notification-based architecture (cont'd)

```
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
}
```



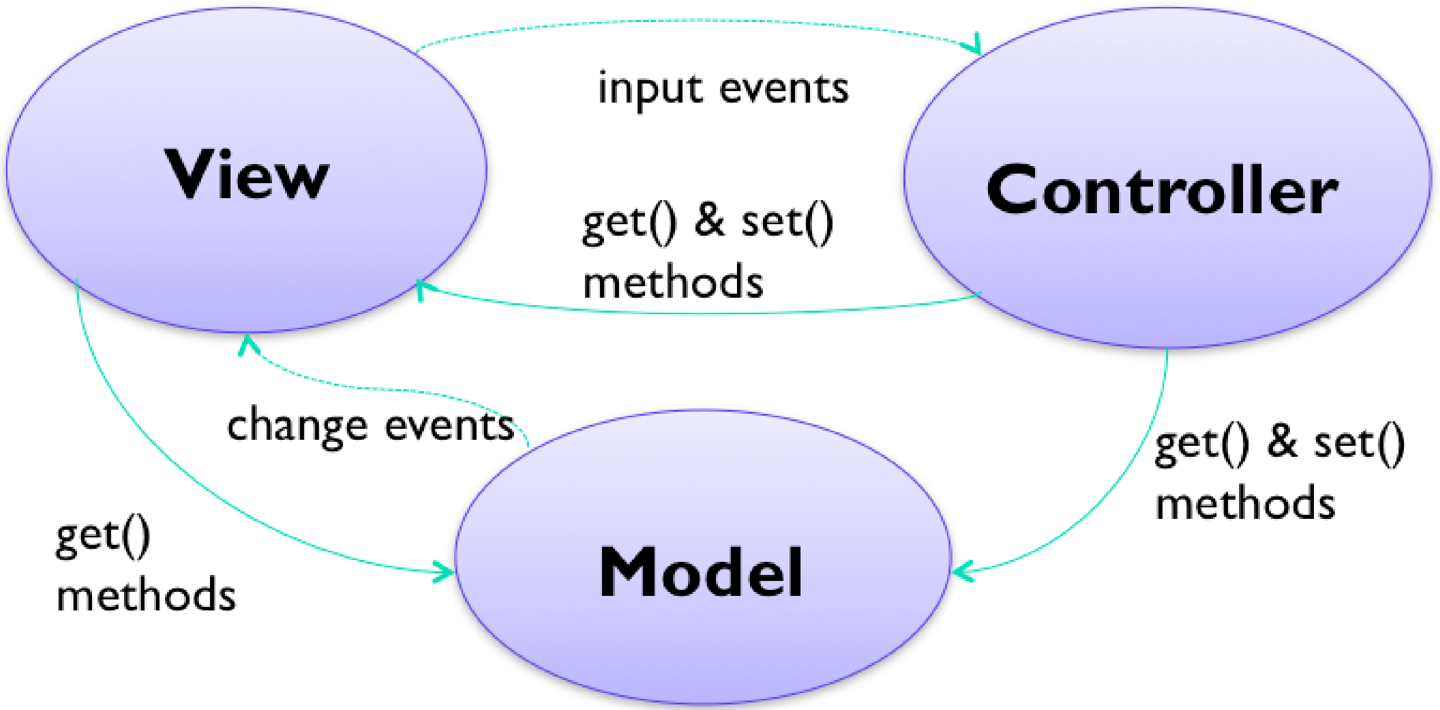
Notification-based architecture (cont'd)

- Main control loop for the event processing is outside the application
- A centralized notifier
  - receives events from the windows system
  - filters events to the application program
- The control flow is **not** controlled by application programmer

Model-View-Controller (MVC)

View handles output
• gets data from the model to display it
• listens for model changes and updates display

Controller handles input
• listens for input events on the view
• calls mutators on model or view

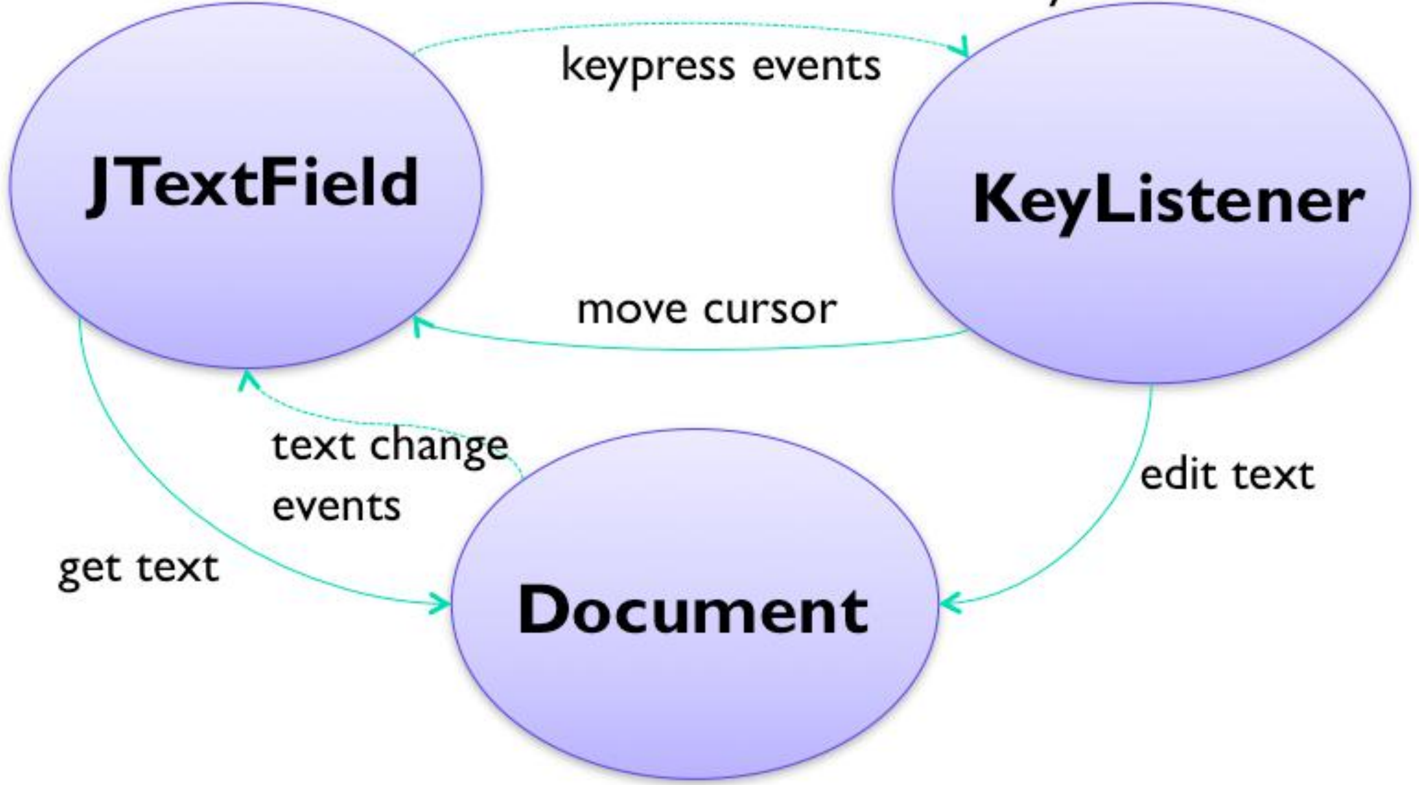


Model maintains application state
• implements state-changing behavior
• sends change events to views

MVC example: textbox

JTextField is a Component that can be added to a view tree

KeyListener is a listener for keyboard events



Document represents a mutable string of characters

MVC: advantages

- Separation of concerns
  - Model: data
  - View: output
  - Controller: input
- Supporting
  - Reuse
    - Reuse of models and views
  - Portability
  - Multiple interfaces
  - Maintainability
    - Changes in view can be done with minimal effects on model

Hard to separate View and Controller

- Controller often needs output
  - View must provide **affordances** for controller
    - e.g., button border and/or icon
  - View must provide feedback about controller state
    - e.g., depressed button
- State shared between controller and view
  - Must be displayed by the view
  - Must be updated and used by the controller
  - Example
    - Checkbox: who manages it when it’s checked and unchecked?

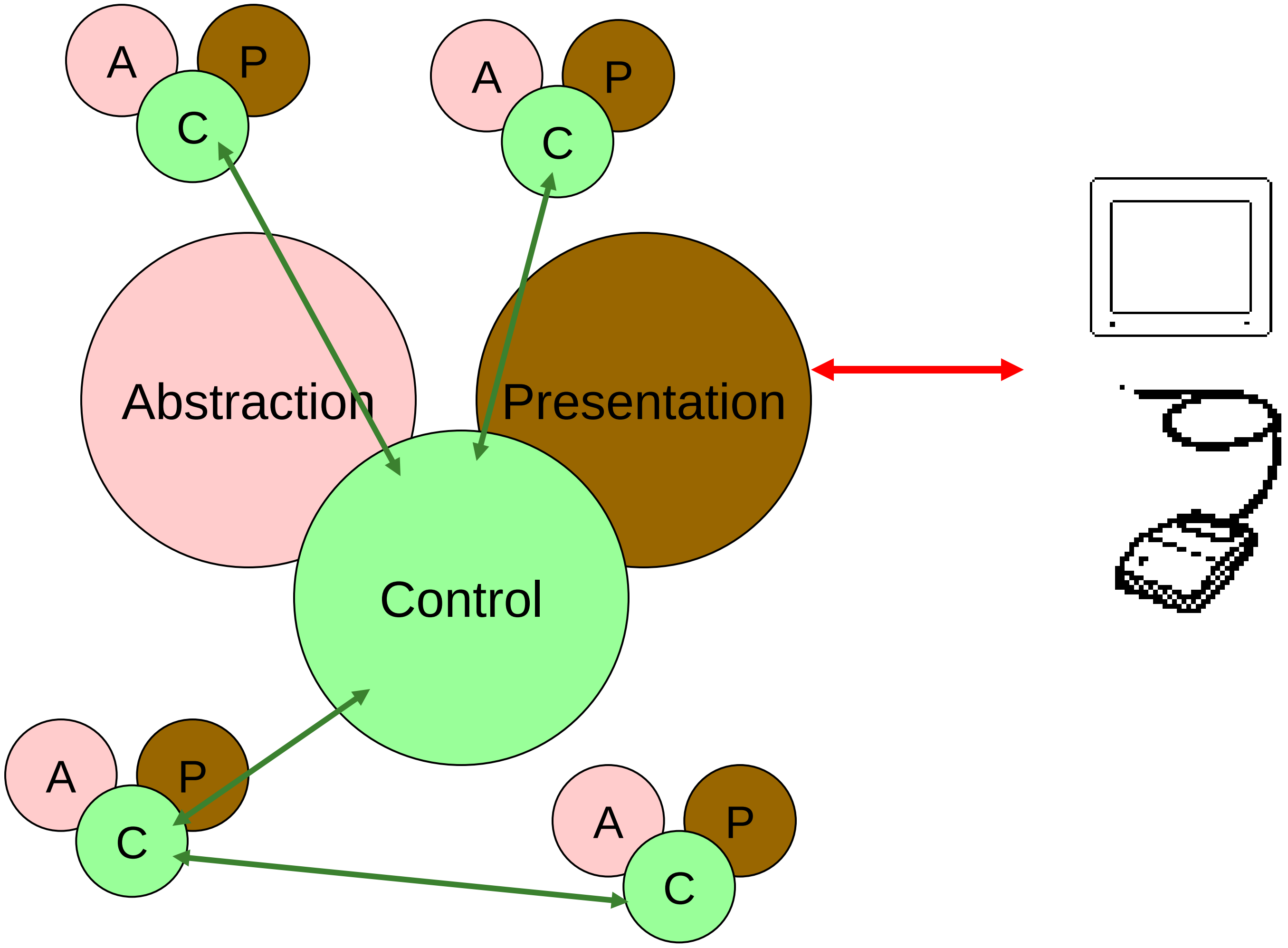
Widget: tightly coupled view and controller

- The MVC idea has largely been replaced by the MV idea
- A widget is a reusable view object that manages both its input and output
  - Also called
    - Components in Java or Flex
    - Controls in Windows
  - Example
    - Menubar
    - Button
    - Editbox

Presentation-Abstraction-Control (PAC)

- A hierarchical structure of agents, each consisting of a triad of Presentation (View), Abstraction (Model), and Control
- Somewhat similar to MVC, but…
  - Completely insulates the Presentation (View in MVC) and the Abstraction
    - provides the option to separately multithread the model and view
    - the user interface (presentation) can be shown before the abstraction has fully initialized

Presentation-Abstraction-Control (PAC)



Outline

- Architectures and design patterns for UIs
- UI development tools

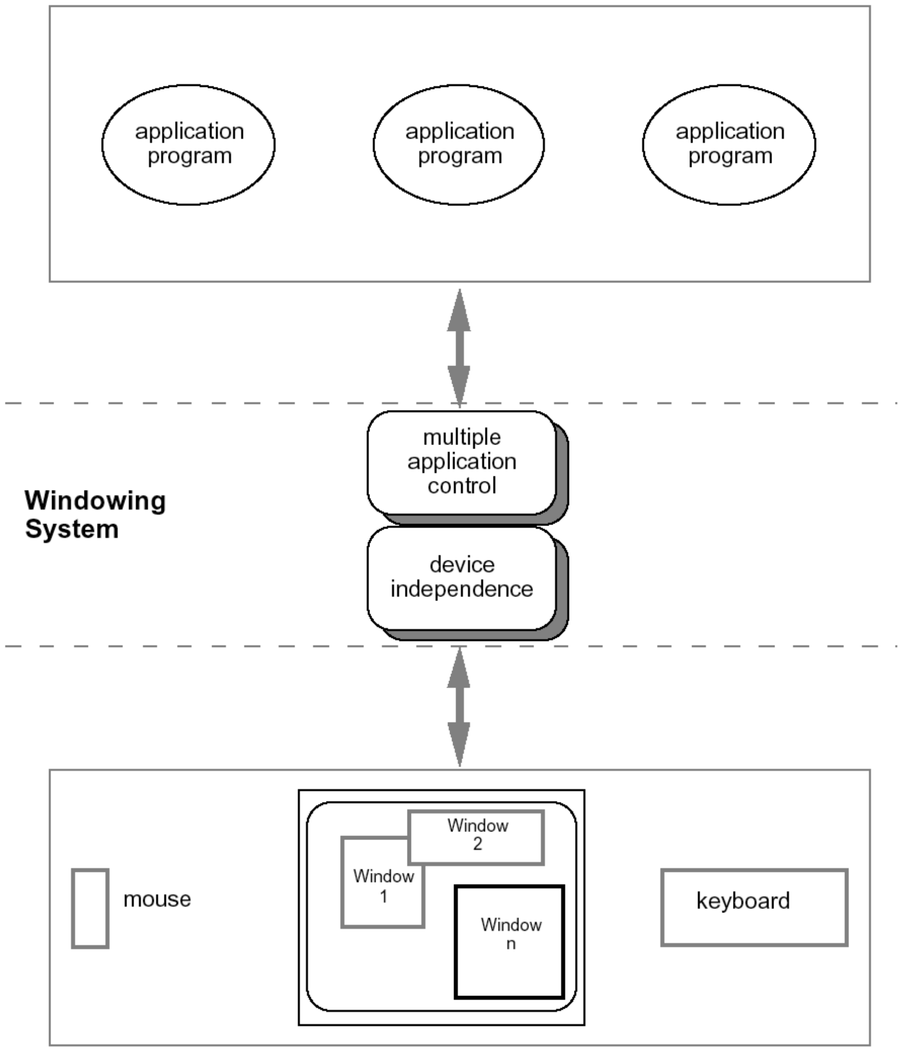
Layers of UI development tools

- Layers of development tools can be broadly categorized into
  - Windowing systems
  - Interface Development Toolkits (IDT)
  - User Interface Management Systems (UIMS)

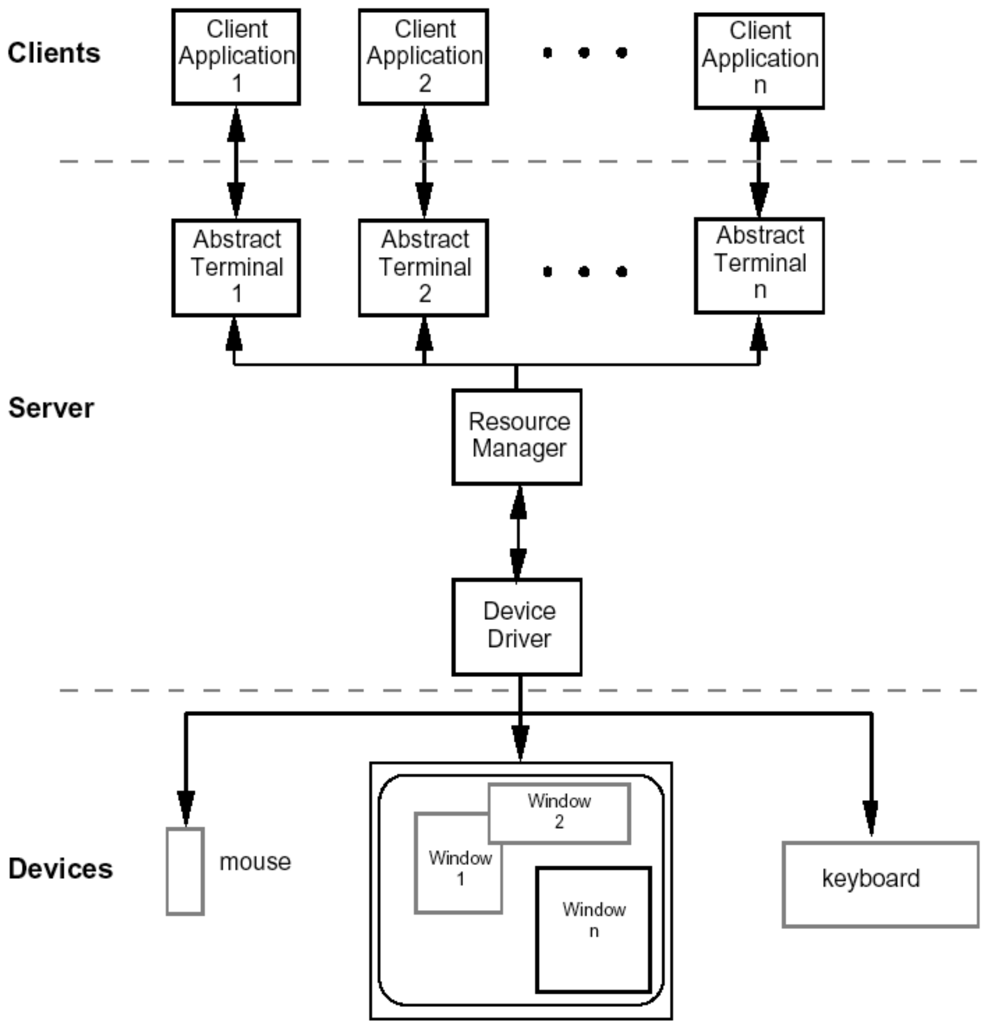
Windowing systems

- Elements
  - Device independence
    - Abstract terminal device drivers
    - Image models for output and (partially) input
      - pixels
      - PostScript (MacOS X, NextStep)
      - Graphical Kernel System (GKS)
  - Multiple application control
    - Simultaneous user tasks
    - Supports independent processes
    - Isolation of individual applications

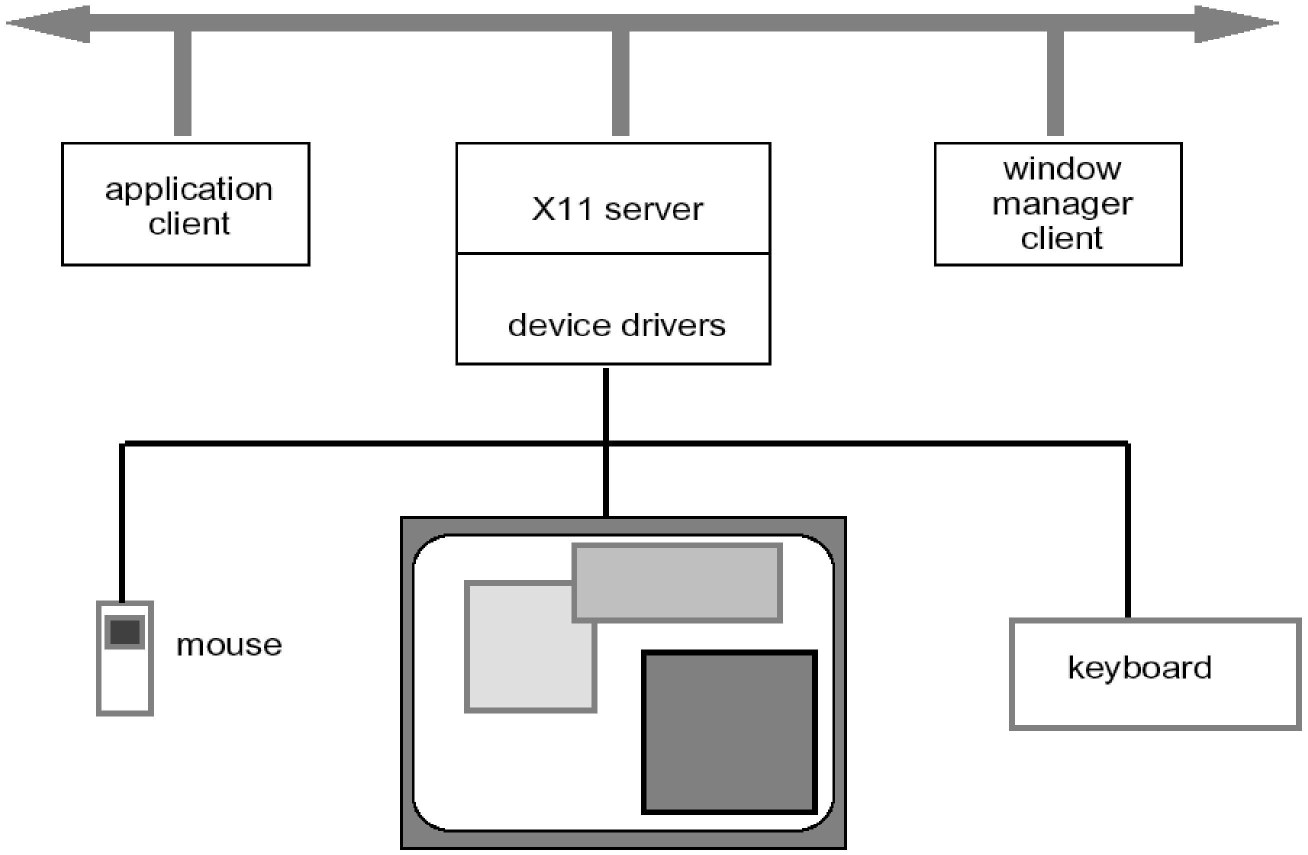
Architectures of windowing system



The Client-Server architecture



X Windows architecture

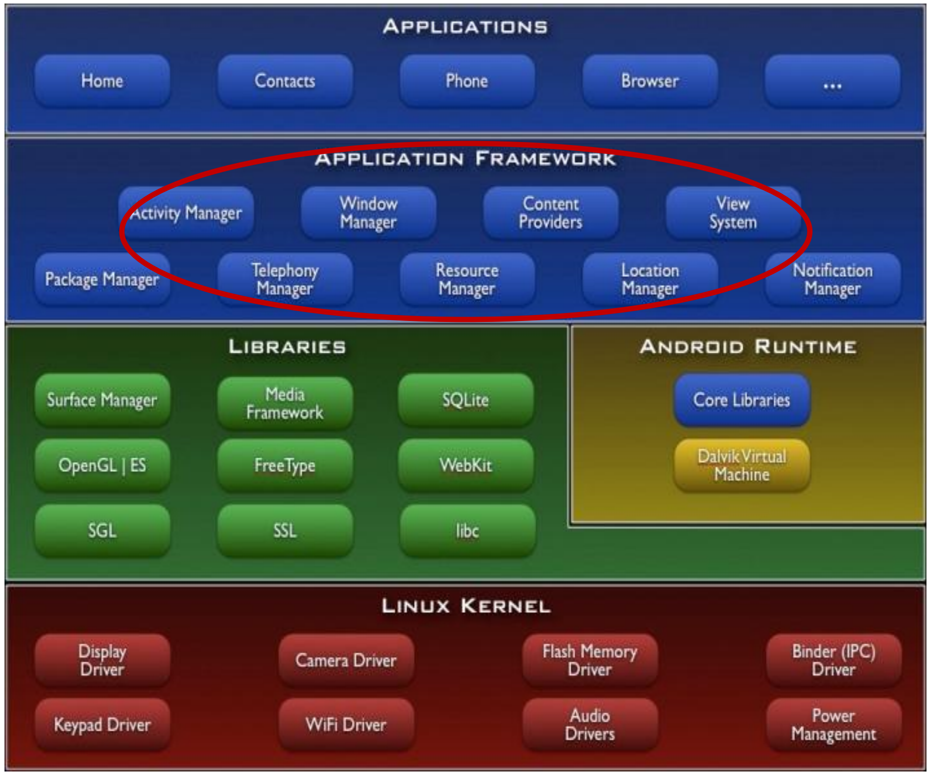


UI development in windowing systems

- UI development tools use **window managers** as the foundation upon which a user interface can be built
  - A window manager allows the user to display, alter, and interact with more than one window at a time
  - The window manager's primary responsibility is to keep track of all aspects of each of the windows being displayed

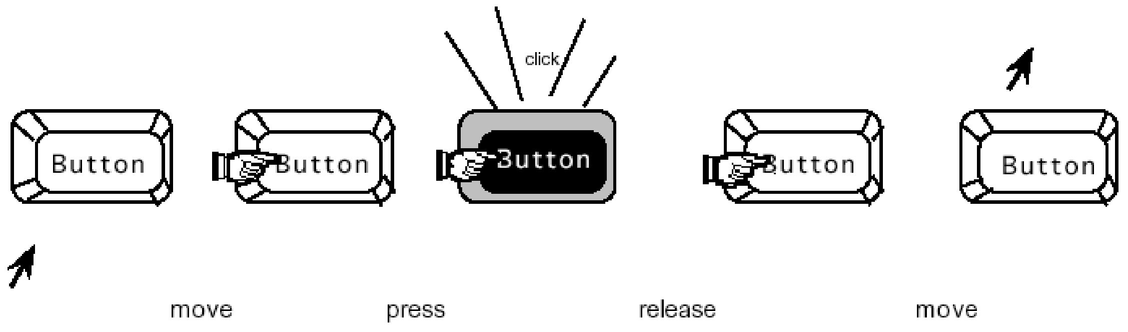
Example: Android

- Google mobile platform
- A software stack for mobile devices
  - OS, middleware, tools, apps
- Core applications
  - email, browser, maps, SMS
- Application Framework
  - Apps and GUI builder
- Linux kernel
- Java programming language



Interface Development Toolkits (IDT)

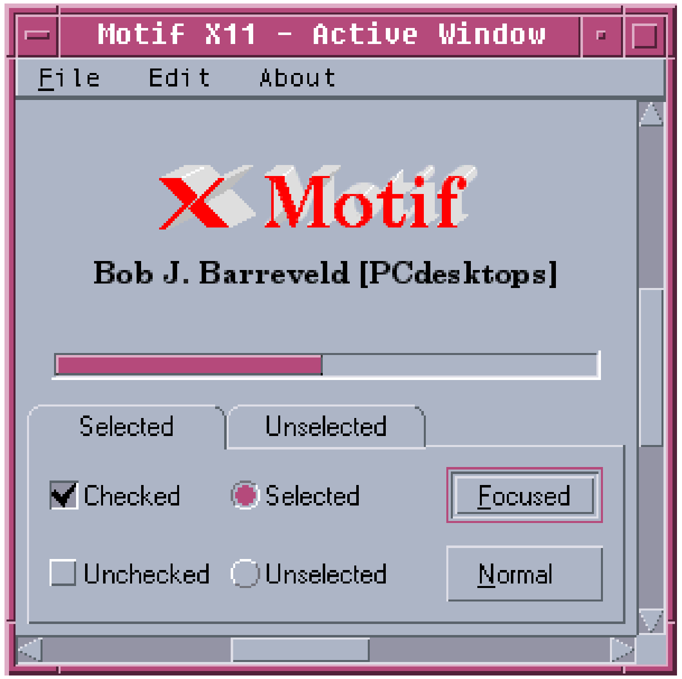
- IDT
  - Provides an interactive editor to layout the interface
  - May provide limited dialogue definition
  - Produces code that represents layout and dialogue (if any)
- Object interaction example
  - Input and output are linked



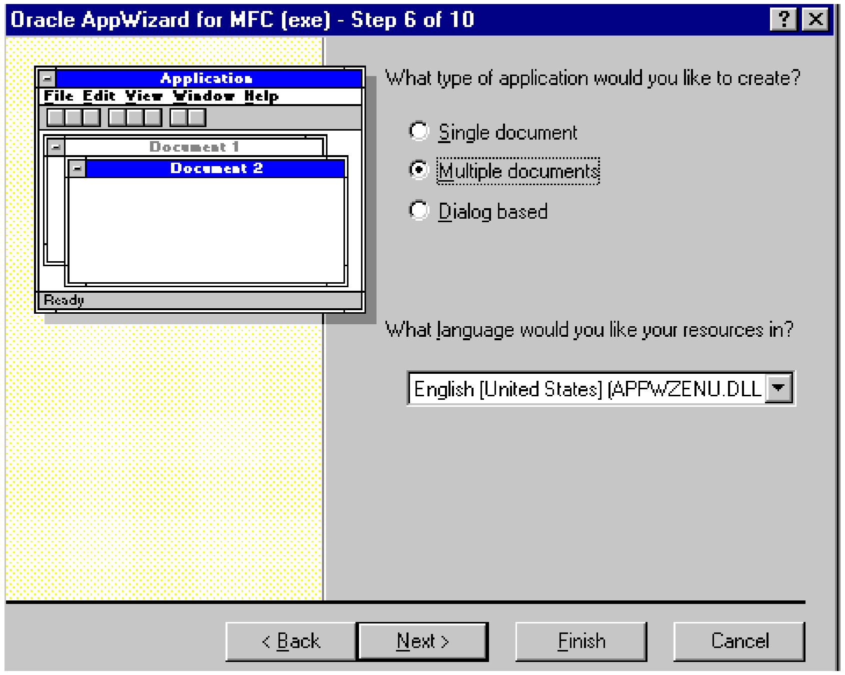
Interface Development Toolkits (IDT)

- Advantages
  - Programming with interaction objects (or techniques, widgets, gadgets)
  - Promote consistency and generalizability
  - Support similar look and feel
  - Similar to object-oriented programming

Interface Development Toolkits (IDT)



Motif for X-window system


Oracle AppWizard for MFC (exe) - Step 6 of 10
Application
File Edit View Window Help
Document 1
Document 2
Ready
< Back
Next >
Finish
Cancel

What type of application would you like to create?

( ) Single document
[✓ Multiple documents]
( ) Dialog based

What language would you like your resources in?

English [United States] (APPWZENU.DLL

MFC

MFC

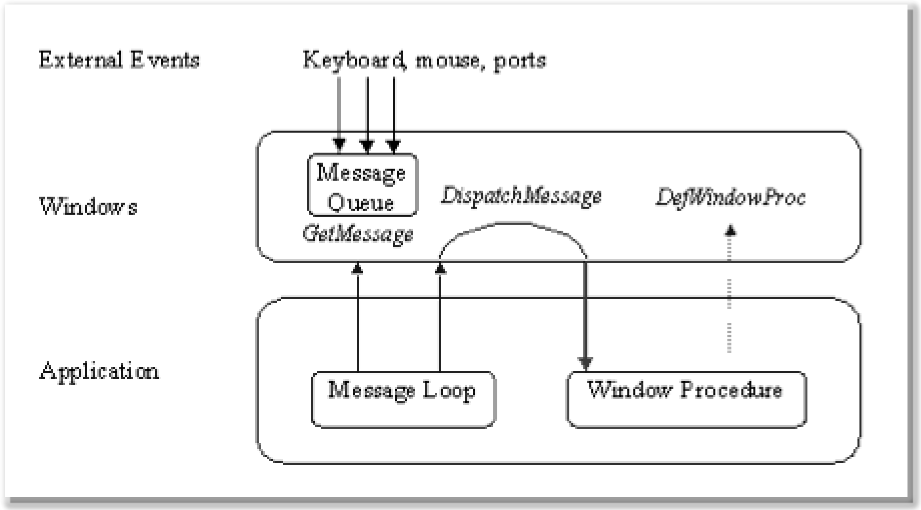
- A library that wraps portions of the Windows API in C++ classes
- Classes are defined for many of the handle-managed Windows objects with predefined windows and controls
- The IDT consists of a software framework used by software developers
  - To promote a standard structure for applications
  - Much simpler to create automatic GUI creation tools when using a standard framework
  - Usually use object-oriented programming (OOP) techniques to implement frameworks

MFC (cont'd)

- When using the **Win32 API**, you will need to
  - Create your core program
  - Create the structures for window and button
  - Initialize all the members for the structures
  - Create callback methods to handle events

→ Programmer has to manage over the entire control and event processing

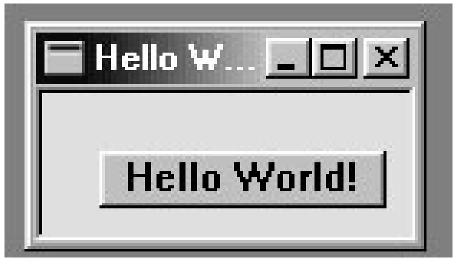
  - very cumbersome task



MFC (cont'd)

- With MFC, this would be easier
  - Two objects in this program
    - One being the Window object
    - Other being the Application object itself
  - HelloWorldWindow :

    public CFrameWnd {…}



  - HelloWorldApplication :

    public CWinApp {…}

Choosing right tools for projects

- Tools can offer large savings in development effort
  - Faster prototyping and development time
  - Less training needed to use tools than program systems
- Different tools need varying levels of expertise
  - Higher level tools need less training
  - Lower level tools offer more control and customization
- Specialized tool exists
  - Tools exist to support specific types of user interfaces
  - Tools available to assist in evaluation, benchmarking, testing

Choosing right tools for projects (cont'd)

- Tools evaluation
  - Using table with tool functionality vs. tool usability

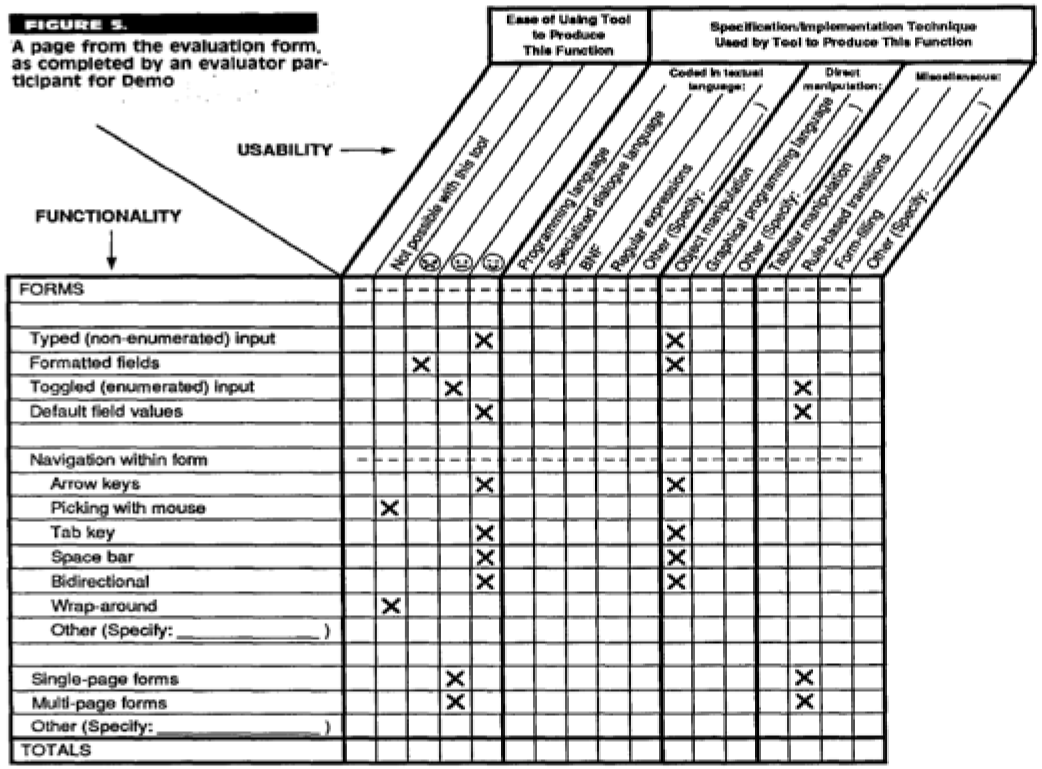


FIGURE 5.
A page from the evaluation form, as completed by an evaluator participant for Demo

USABILITY →

FUNCTIONALITY ↓

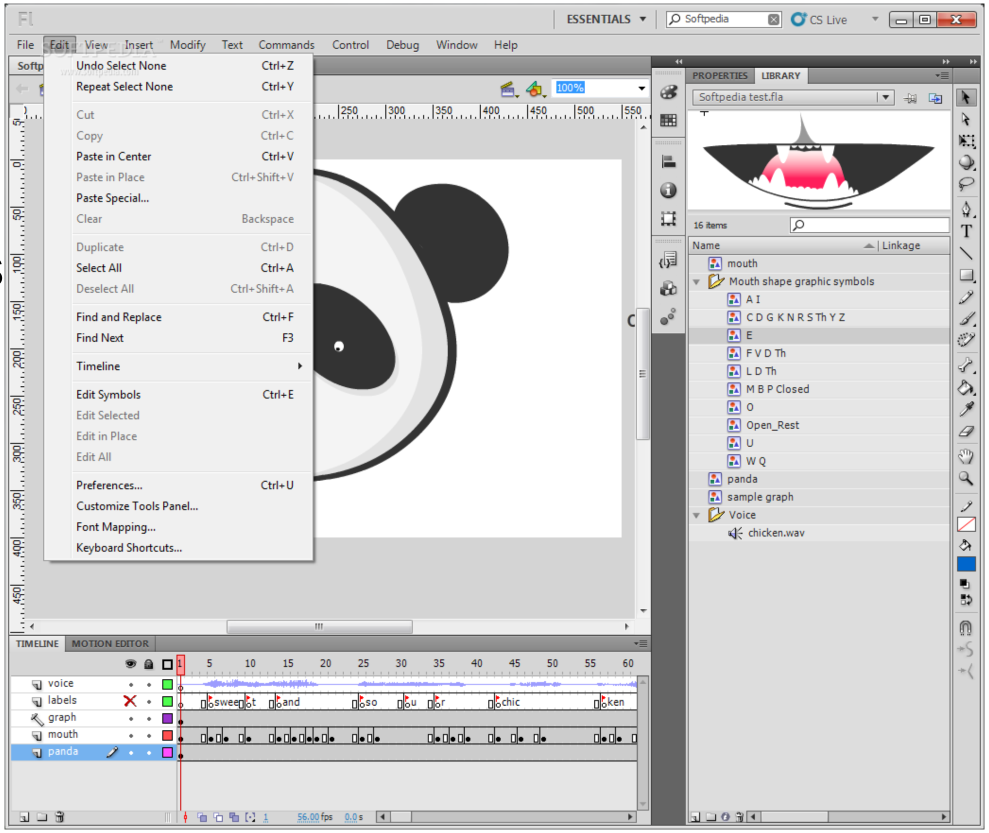
	Ease of Using Tool to Produce This Function					Specification/Implementation Technique Used by Tool to Produce This Function											
						Coded in textual language:					Direct manipulation:			Miscellaneous:			
		Not possible with this tool	☹	😐	☺	Programming language	Specialized dialogue language	BNF	Regular expressions	Other (Specify: ____)	Object manipulation	Graphical programming language	Other (Specify: ____)	Tabular manipulation	Rule-based transitions	Form-filling	Other (Specify: ____)
FORMS																	
Typed (non-enumerated) input					X						X						
Formatted fields			X								X						
Toggled (enumerated) input				X											X		
Default field values					X										X		
Navigation within form																	
Arrow keys					X						X						
Picking with mouse		X															
Tab key					X						X						
Space bar					X						X						
Bidirectional					X						X						
Wrap-around		X															
Other (Specify: ____________)																	
Single-page forms				X											X		
Multi-page forms				X											X		
Other (Specify: ____________)																	
TOTALS																	

(from D. Hix and R. Schulman, CACM, pp.79, March 1991/Vo1.34, No.3)

Common tools

Flash (Adobe)

- Shockwave Flash and Macromedia Flash
- Supports vector/raster graphics animations, bidirectional
- video/audio streams
- Contains scripting language (ActionScript)
- A browser plug-in, Flash player
- Good for concept prototyping



Common tools (cont'd)

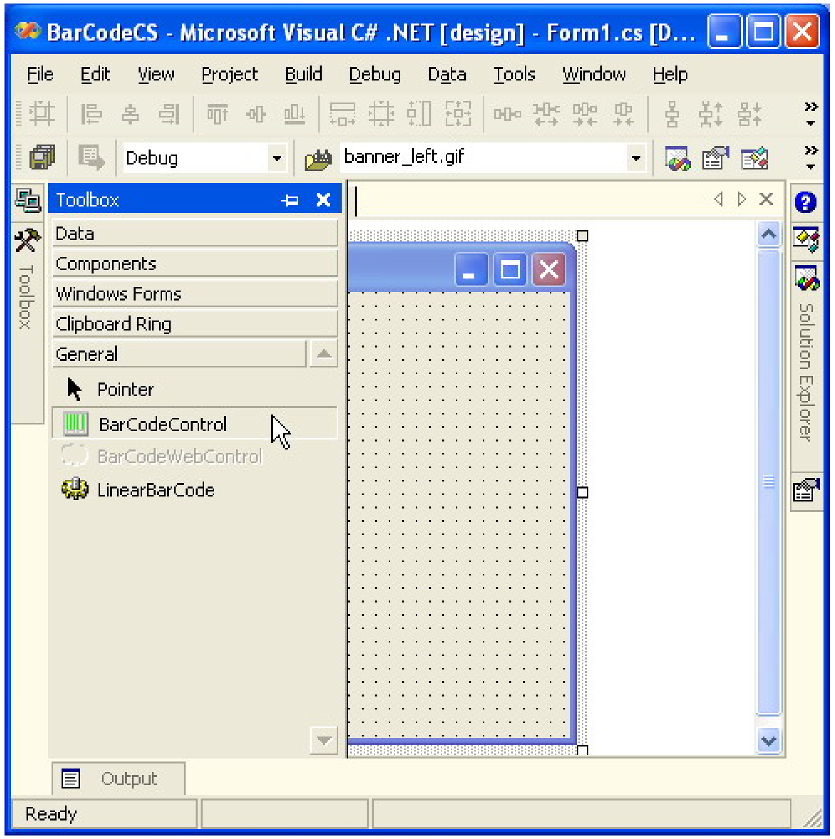
Java toolkits

- Java toolkit – AWT (Abstract Window Toolkit)
  - an Abstract Window Toolkit, part of the Java Foundation Classes (JFC)
  - Standard API for providing GUIs for Java programs
  - Notification based architecture
  - AWT 1.0 – needs to subclass basic widgets
  - AWT 1.1 and beyond -– callback objects
- Swing toolkit
  - built on top of AWT – higher level features
  - provide a more sophisticated set of GUI components than AWT
  - uses MVC architecture

Common tools (cont'd)

Windows .NET

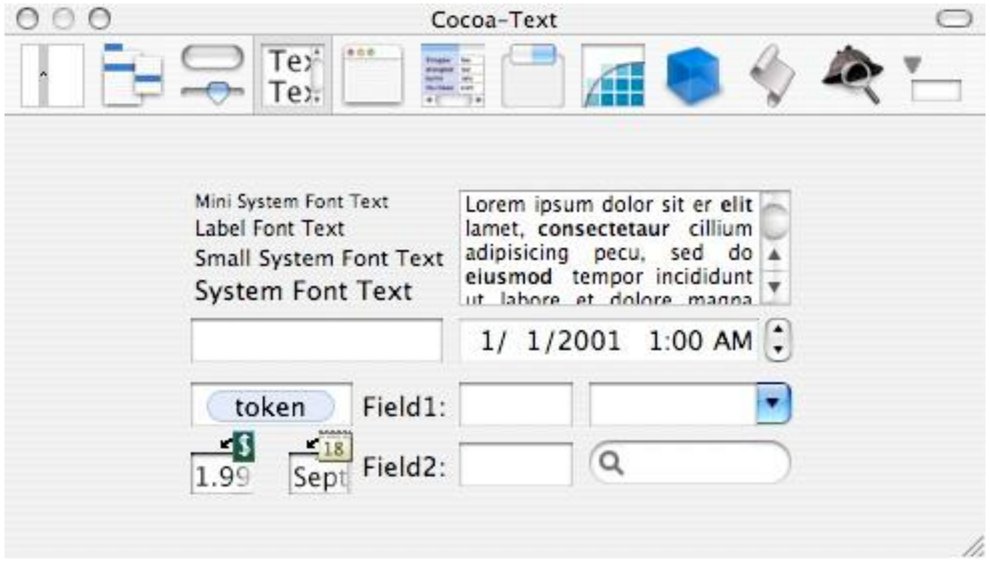
- Windows GUI builder
  - integral components
- Form-Control pattern
- Element drag-and-drop
- Windows themes support
- OO programming environment
- Visual Studio tools



Common tools (cont'd)

Apple Cocoa

- Mac OS-X GUI builder
- Xcode & Interface Builder
- MVC architecture
  - Cocoa bindings to reduce dependencies between models, views and controllers
- Objective-C, also supports Java, Python, Perl



Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna

Common tools (cont’d)

Ruby on Rails

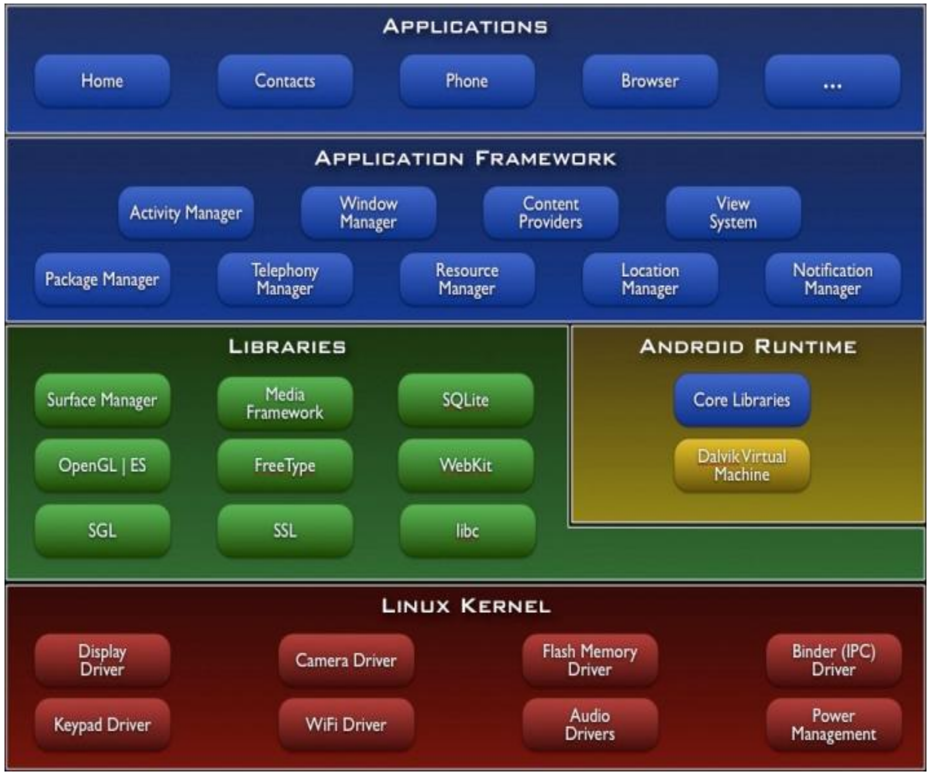
- An open-source web framework
- Good for developing database backed web applications
- Convention over configuration
- MVC pattern
  - MySQL DB (Model)
  - Browser-based (View)
  - Ruby (Controller)

http://www.rubyonrails.org/

Common tools (cont'd)

Android

- Google mobile platform
- A software stack for mobile devices
  - OS, middleware, tools, apps
- Core applications
  - email, browser, maps, SMS
- Application Framework
  - Apps and GUI builder
- Linux kernel
- Java programming language



Common tools (cont'd)

S60/ Symbian Platform

- SF platform for Symbian
- OS based mobile phone
- Nokia, Samsung, LG, Lenovo
- Includes standard apps and GUI builder
- Structured MVC pattern
- Supports Java, Symbian C++, Python

Tool resources on the Internet

- A collection of UI tools (no longer maintaining)
  - http://www.cs.cmu.edu/afs/cs/user/bam/www/toolnames.html
- 20 useful tools for web development
  - http://sixrevisions.com/tools/20_web_development_tools/
- 24 usability testing tools
  - http://www.usefulusability.com/24-usability-testing-tools/

Tool terminologies

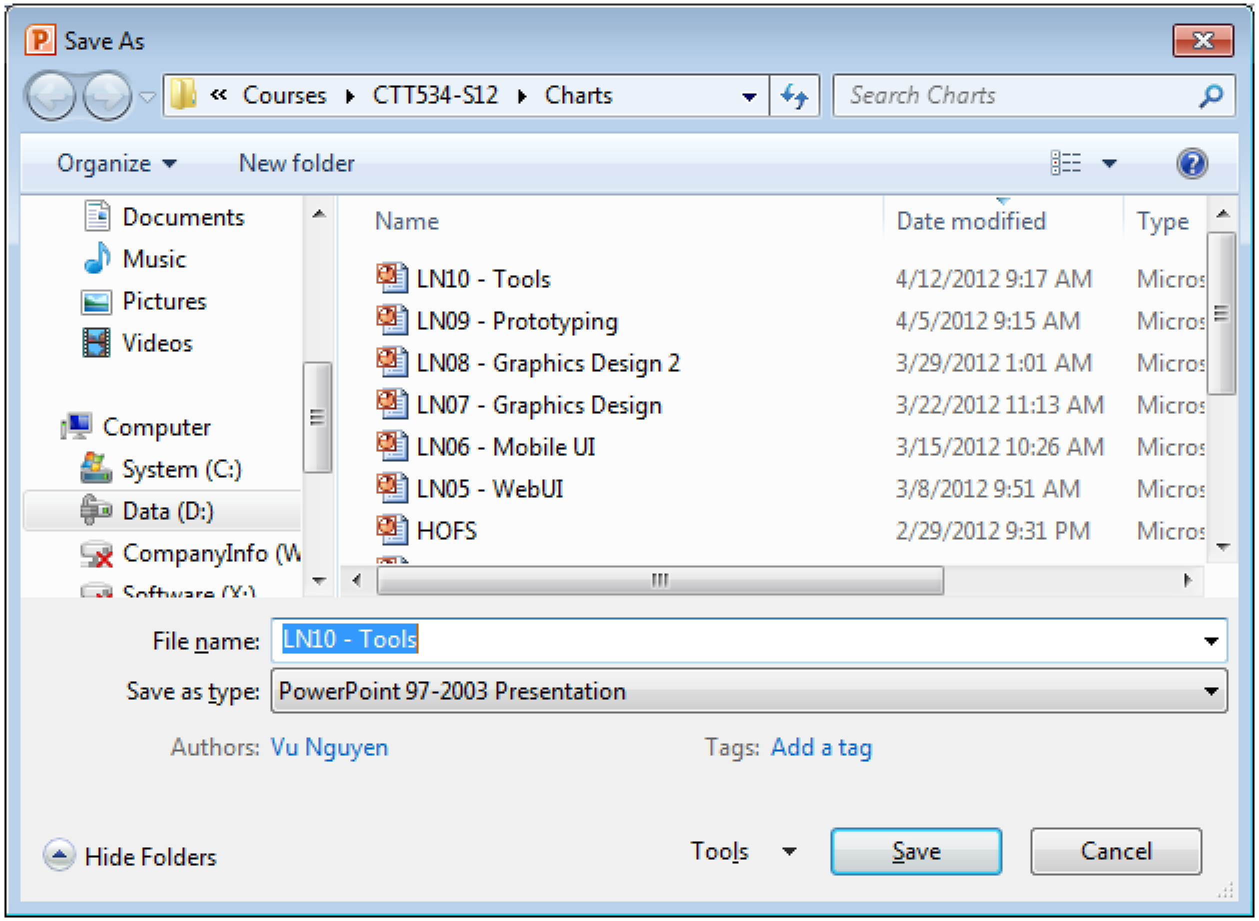
- Presentation – that part of the software that displays objects
  - gets inputs from user and modifies screens
- Dialogue – describes the dynamic behavior of the interface
  - what happens when button is pressed
  - has code to determine next objects to display or modify
- Lay out – describes what the visual part will look like
  - contains information on color, size, etc.
  - does not describe any dynamic behavior
- Dialogue/Domain Interface
  - exchanges data between application and dialogue
  - causes application and interface to be independent
- Domain – the core of the application
  - performs non-interface calculations
  - should have no knowledge of how data is displayed

Tool terminologies (cont'd)

- User Interface Management System (UIMS)
  - Provides mechanism for separating the user interface & application
  - Contains language for defining dialogue interface
- Interface Design Tool (IDT)
  - Provides an interactive editor to layout the interface
  - May provide limited dialogue definition
  - Produces code that represents layout and dialogue (if any)
- Specialized Languages
  - High level languages that abstracts away windowing system
- Layout Languages
  - Languages used to define position and attribute of interface objects
- Virtual Toolkits
  - Toolkits that are look and feel independent
  - Allows application to be developed once & moved between platforms
- Toolkits
  - Provide objects of user interface that define look and feel

UI Hall of Fame or Shame

- Model dialog



Save As

« Courses ▸ CTT534-S12 ▸ Charts

Search Charts

Organize ▾ New folder

Documents
Music
Pictures
Videos
Computer
System (C:)
Data (D:)
CompanyInfo (W
Software (X:)

Name	Date modified	Type
LN10 - Tools	4/12/2012 9:17 AM	Micros
LN09 - Prototyping	4/5/2012 9:15 AM	Micros
LN08 - Graphics Design 2	3/29/2012 1:01 AM	Micros
LN07 - Graphics Design	3/22/2012 11:13 AM	Micros
LN06 - Mobile UI	3/15/2012 10:26 AM	Micros
LN05 - WebUI	3/8/2012 9:51 AM	Micros
HOFS	2/29/2012 9:31 PM	Micros

File name: LN10 - Tools

Save as type: PowerPoint 97-2003 Presentation

Authors: Vu Nguyen

Tags: Add a tag

Hide Folders

Tools ▾ Save Cancel