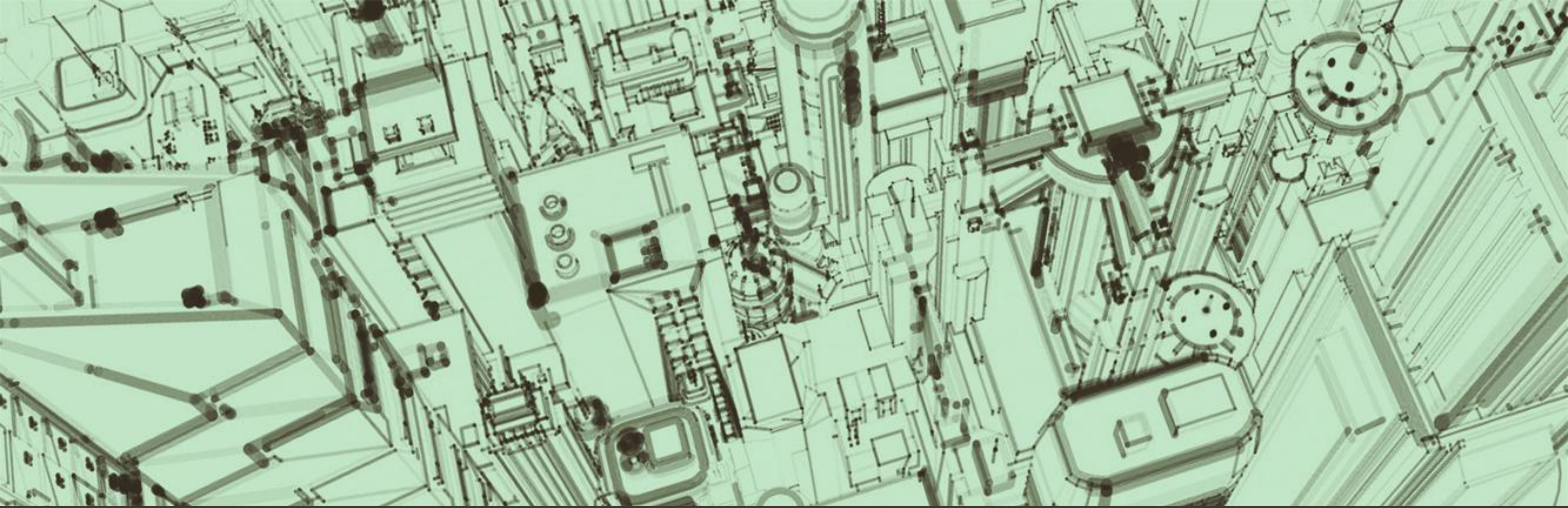


Software Engineering

Sobia Iftikhar
Sobia.Iftikhar@nu.edu.pk

Week 09

Class 22

07-APRIAL-2021

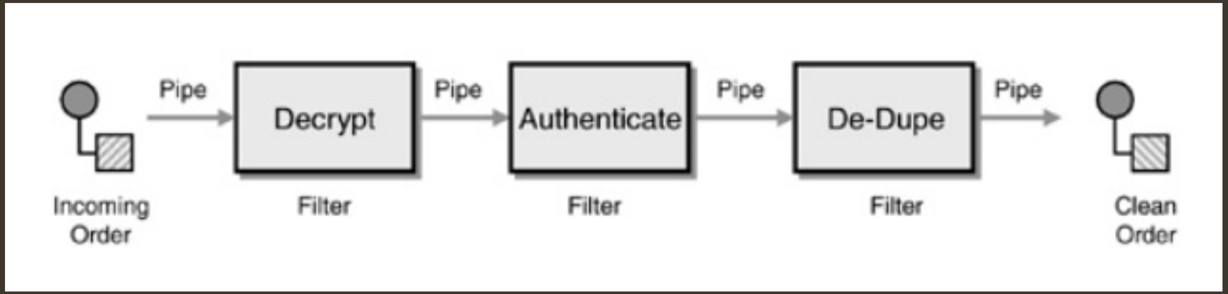
Content

- Architectural design decisions
- Architectural views
- Architectural patterns
- Application architectures

Pipe & Filter Architecture

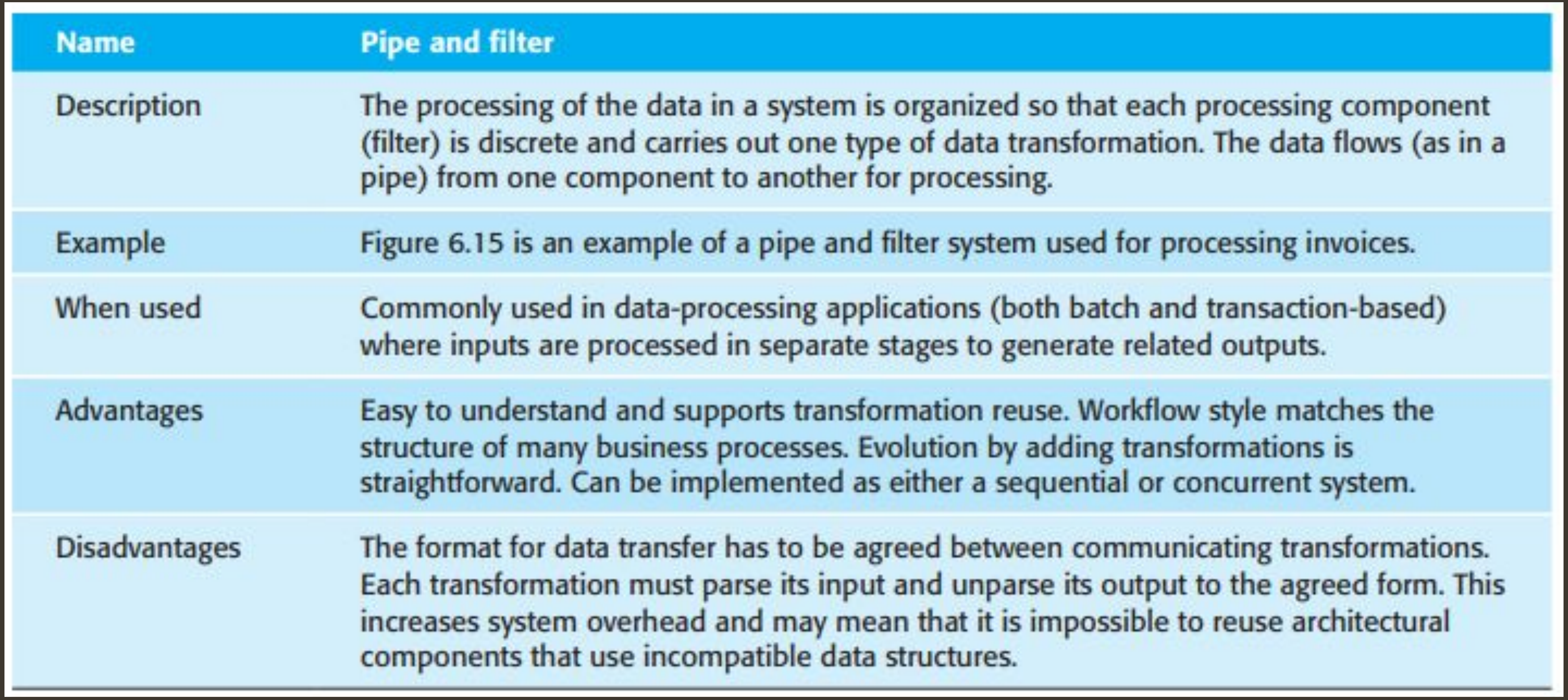
- Pipe and Filter is another architectural pattern.
- which has independent entities called filters (components) which perform transformations on data and process the input they receive,
- and pipes, which serve as connectors for the stream of data being transformed, each connected to the next component in the pipeline.
- Many systems are required to transform streams of discrete data items, from input to output.
- Many types of transformations occur repeatedly in practice, and so it is desirable to create these as independent, reusable parts, Filters

Example



- Compilers:
  Lexical Analysis -> parsing -> semantic analysis -> code generation
- Signal Processing

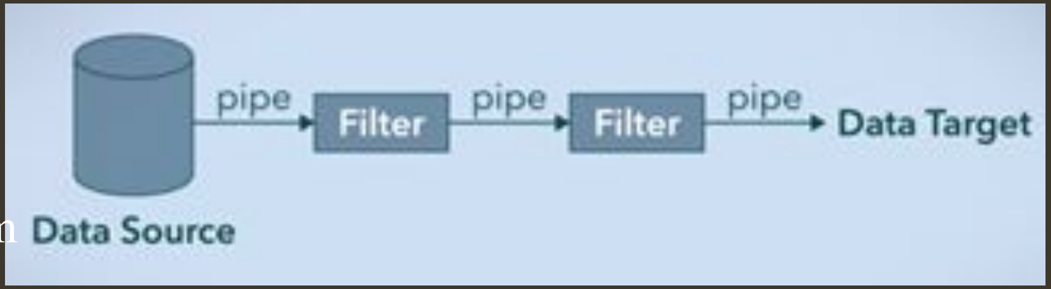
Pattern Name



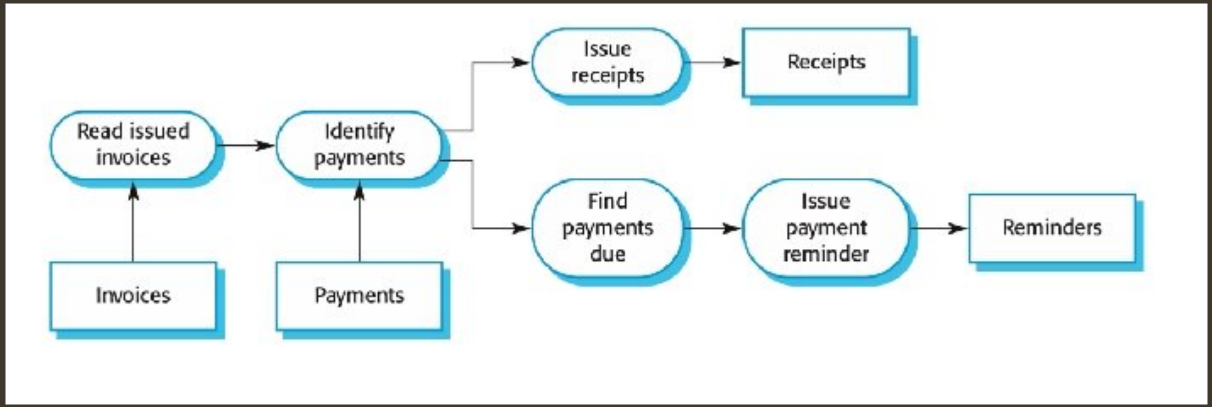
Name	Pipe and filter
Description	The processing of the data in a system is organized so that each processing component (filter) is discrete and carries out one type of data transformation. The data flows (as in a pipe) from one component to another for processing.
Example	Figure 6.15 is an example of a pipe and filter system used for processing invoices.
When used	Commonly used in data-processing applications (both batch and transaction-based) where inputs are processed in separate stages to generate related outputs.
Advantages	Easy to understand and supports transformation reuse. Workflow style matches the structure of many business processes. Evolution by adding transformations is straightforward. Can be implemented as either a sequential or concurrent system.
Disadvantages	The format for data transfer has to be agreed between communicating transformations. Each transformation must parse its input and unparse its output to the agreed form. This increases system overhead and may mean that it is impossible to reuse architectural components that use incompatible data structures.

Example

The name “pipe and filter” comes from the original Unix system where it was possible to link processes using “pipes.”



Pipe and filter systems are best suited to batch processing systems and embedded systems where there is limited user interaction.



Application Architecture

- Application systems are intended to meet a business or an organizational need. All businesses have much in common—they need to hire people, issue invoices, keep accounts, and so on.
- These commonalities have led to the development of software architectures that describe the structure and organization of particular types of software systems.
- For example, in real-time systems, there might be generic architectural models of different system types, such as data collection systems or monitoring systems. Although instances of these systems differ in detail.
- The application architecture may be reimplemented when developing new systems.

you can use models of application architectures in a number of ways:

- For example, a system for supply chain management can be adapted for different types of suppliers, goods, and contractual arrangements

•As a starting point for the architectural design process As a design checklist

•As a way of organizing the work of the development team

•As a means of assessing components for reuse.

•As a vocabulary for talking about applications

- As a starting point for the architectural design process As a design checklist
  - If you are unfamiliar with the type of application that you are developing, you can base your initial design on a generic application architecture.
- As a means of assessing components for reuse.
  - If you have components you might be able to reuse, you can compare these with the generic structures to see whether there are comparable components in the application architecture.
- As a way of organizing the work of the development team
  - The application architectures identify stable structural features of the system architectures
- As a vocabulary for talking about applications
  - If you are discussing a specific application or trying to compare applications

Example of Application Architecture

- **Data processing applications**
  - Data driven applications that process data in batches without explicit user intervention during the processing. E.g scientific data processing, commercial data processing and data analysis.
- **Transaction processing applications**
  - Data centered applications that process user requests and update information in a system database. E.g. E-commerce processing, bank transfers
- **Event processing systems**
  - Applications where system actions depend on interpreting events from the system’s environment. E.g wireless sensor applications
- **Language processing systems**
  - Applications where theusers’intentions are specified in a formal language that is processed and interpreted by the system. E.gNatural Language processing

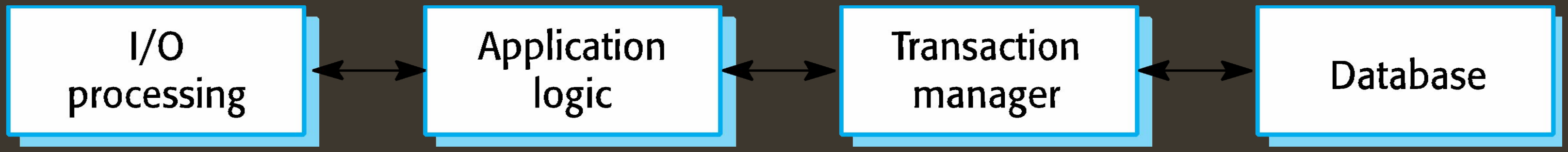
Application type examples

- Two very widely used generic application architectures are transaction processing systems and language processing systems.
- Transaction processing systems
  - E-commerce systems;
  - Reservation systems.
- Language processing systems
  - Compilers;
  - Command interpreters.

Transaction processing systems

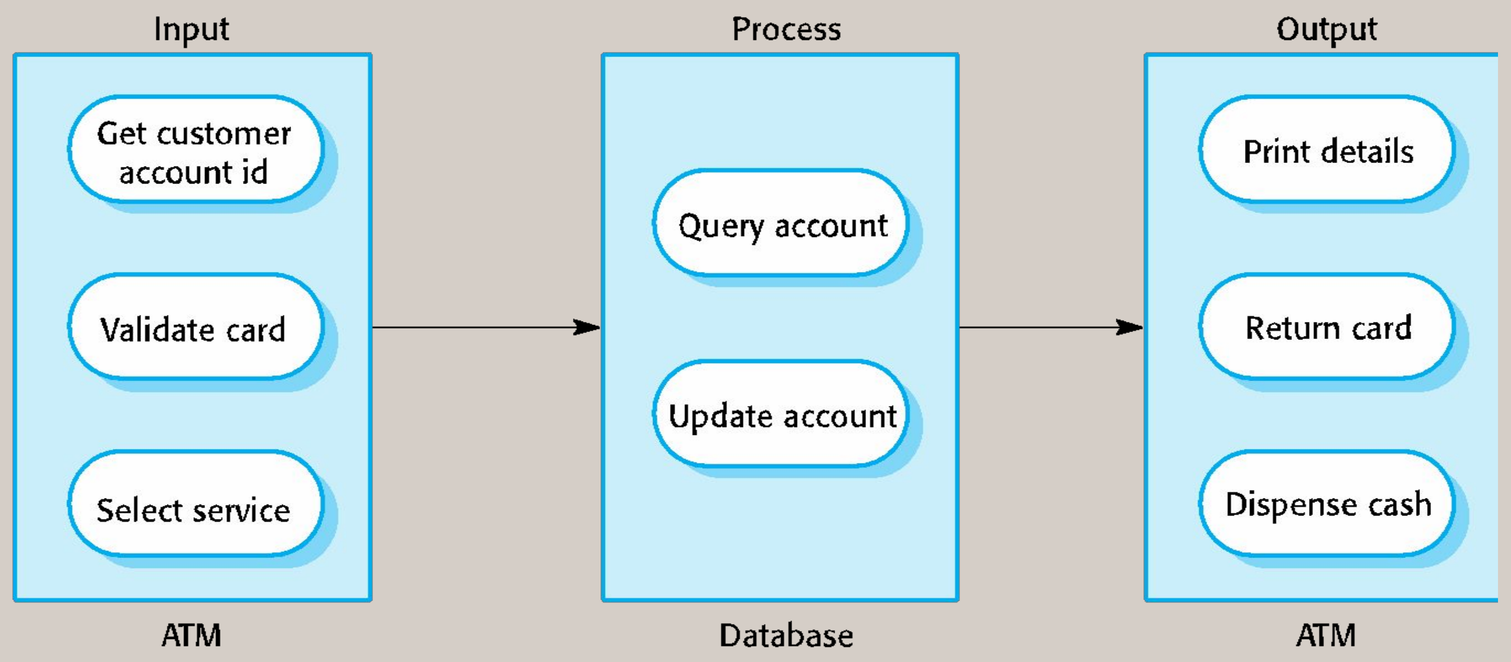
- Process user requests for information from a database or requests to update the database.
- They are organized in such a way that user actions can’t interfere with each other and the integrity of the database is maintained.
- From a user perspective a transaction is:
  - Any coherent sequence of operations that satisfies a goal;
  - For example - find the times of flights from London to Paris.
    - interactive banking systems, e-commerce systems, information systems, and booking systems.
- Users make asynchronous requests for service which are then processed by a transaction manager.

The structure of transaction processing applications



Transactions
Processing
LOADING...
Reporting
Data Capture
Updating

The software architecture of an ATM system



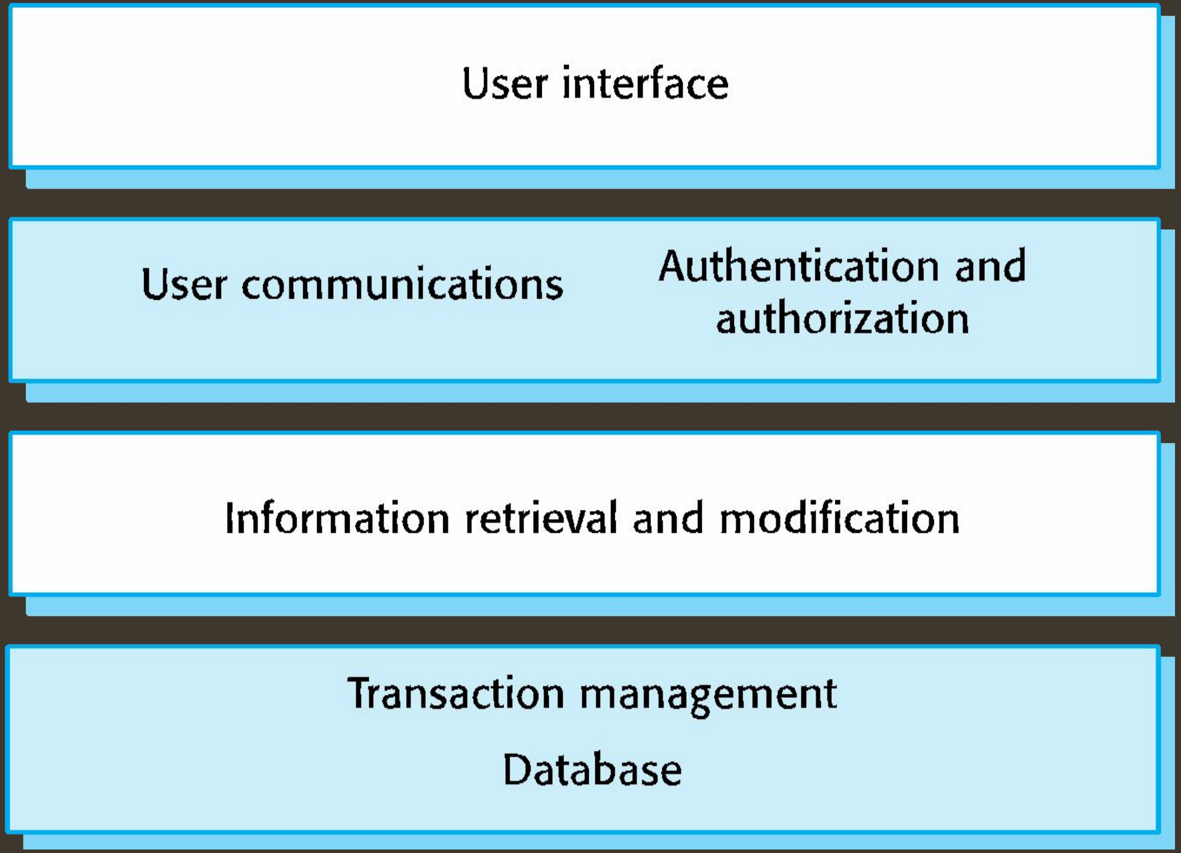
Information systems architecture

- Information systems are almost always web-based systems, where the user interface is implemented in a web browser.
- All systems that involve interaction with a shared database can be considered to be transaction-based information systems.
- such as a library catalog, a flight timetable, or-the records of patients in a hospital. Information

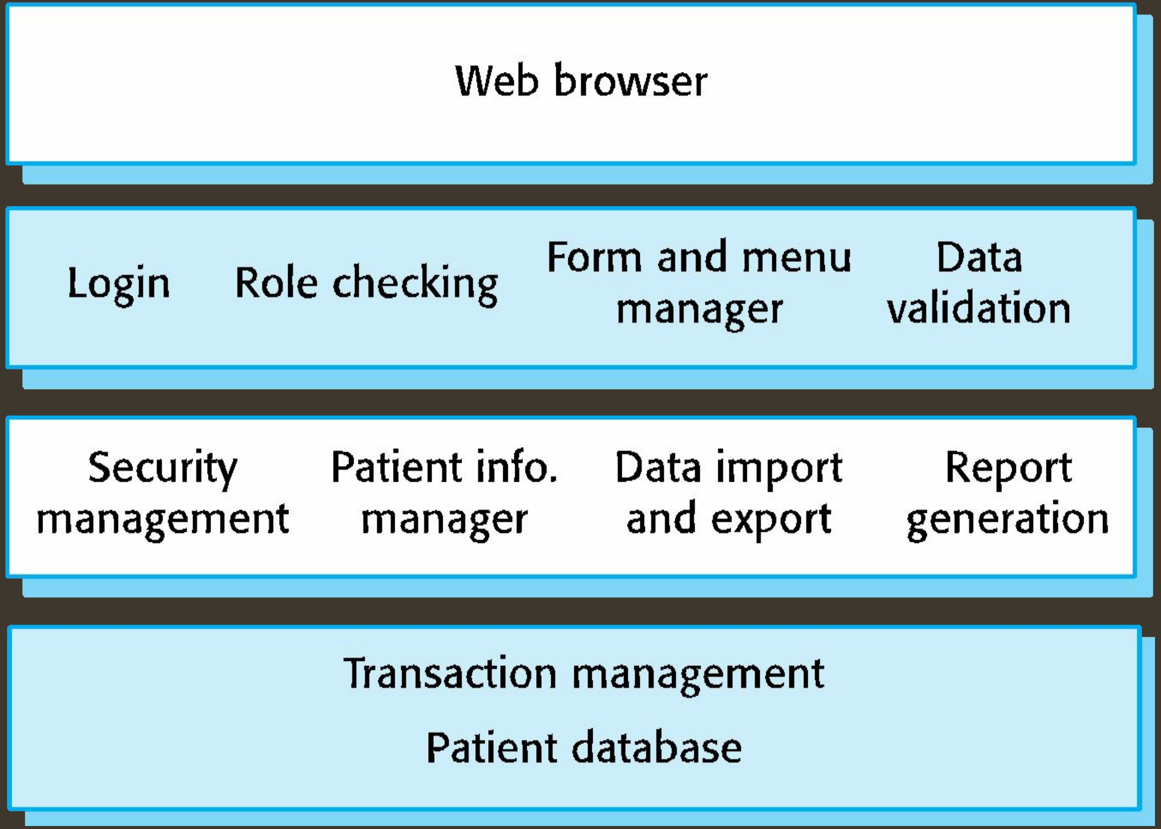
Information systems architecture

- Information systems have a generic architecture that can be organized as a layered architecture.
- Layers include:
  - The user interface
  - User communications
  - Information retrieval
  - System database

Layered information system architecture


User interface
User communications
Authentication and authorization
Information retrieval and modification
Transaction management
Database

The architecture of the Mentcare system



Web-based information systems

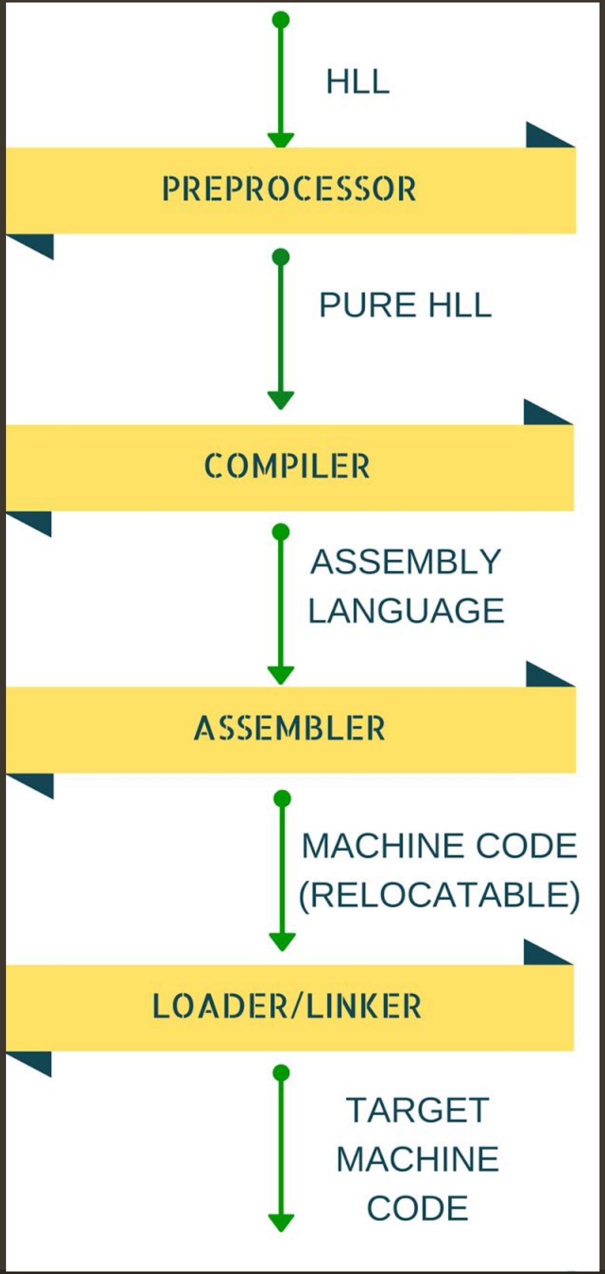
- Information and resource management systems are now usually web-based systems where the user interfaces are implemented using a web browser.
- For example, e-commerce systems are Internet-based resource management systems that accept electronic orders for goods or services and then arrange delivery of these goods or services to the customer.
- In an e-commerce system, the application-specific layer includes additional functionality supporting a ‘shopping cart’ in which users can place a number of items in separate transactions, then pay for them all together in a single transaction.

Server implementation

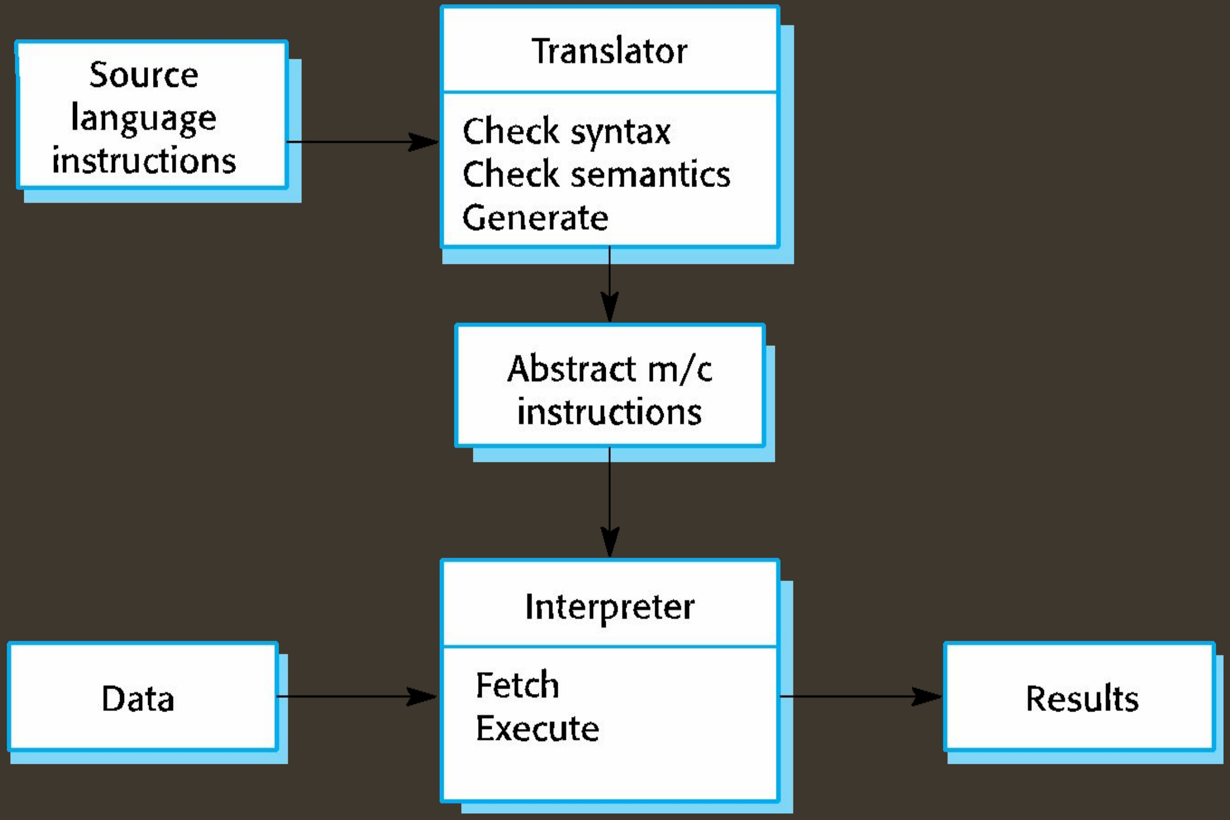
- These systems are often implemented as multi-tier client server/architectures (discussed in Chapter 17)
  - The web server is responsible for all user communications, with the user interface implemented using a web browser;
  - The application server is responsible for implementing application-specific logic as well as information storage and retrieval requests;
  - The database server moves information to and from the database and handles transaction management.

Language processing systems

- Language processing systems are systems in which the user’s intentions are expressed in a formal language, such as a programming language.
- The language processing system processes this language into an internal format and then interprets this internal representation. T
- he best known language processing systems are compilers,
- which translate high-level language programs into machine code.
- However, language processing systems are also used to interpret command languages for databases and information systems, and markup languages such as XML.


HLL
PREPROCESSOR
PURE HLL
COMPILER
ASSEMBLY LANGUAGE
ASSEMBLER
MACHINE CODE (RELOCATABLE)
LOADER/LINKER
TARGET MACHINE CODE

The architecture of a language processing system



Compiler components

- A lexical analyzer, which takes input language tokens and converts them to an internal form.
- A symbol table, which holds information about the names of entities (variables, class names, object names, etc.) used in the text that is being translated.
- A syntax analyzer, which checks the syntax of the language being translated.
- A syntax tree, which is an internal structure representing the program being compiled.

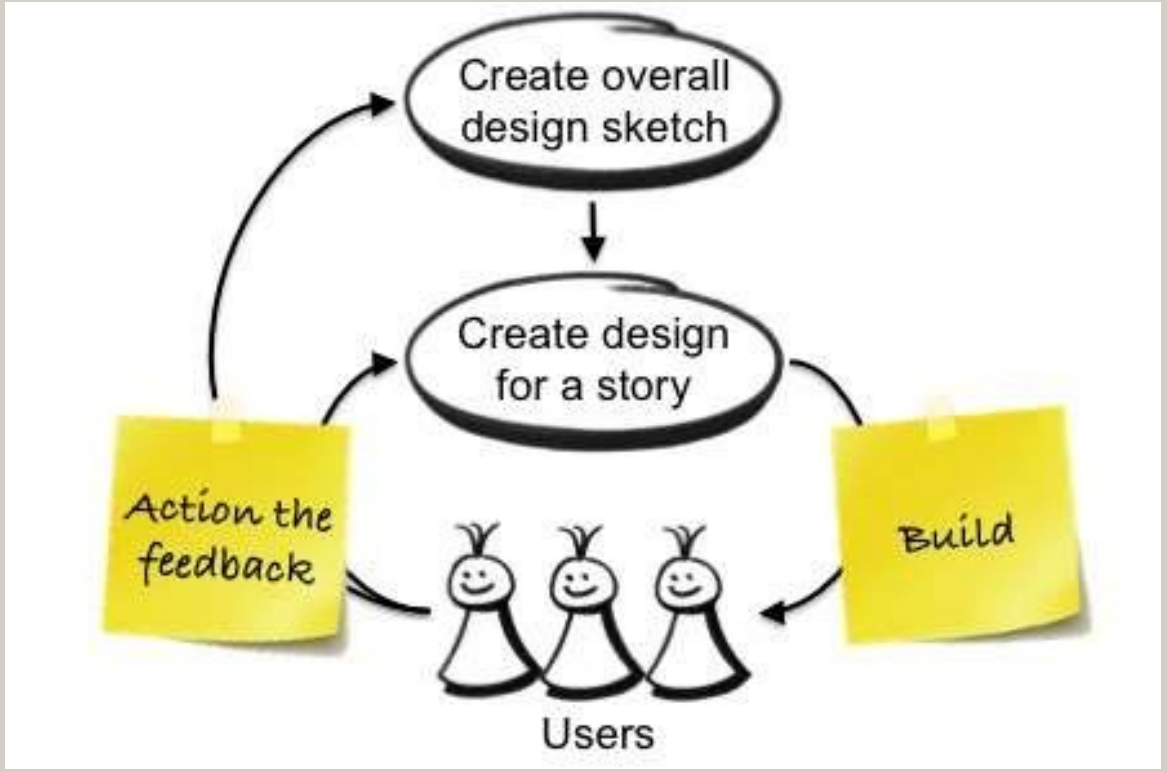
Class 23

08-APRIAL-2021

Chapter #11
User Interface Design

- The Golden Rules
  - Place the User in Control
  - Reduce the User’s Memory Load
  - Make the Interface Consistent
- User Interface Analysis and Design
  - Interface Analysis and Design Models
  - The Process
- Interface Analysis
  - User Analysis
  - Task Analysis and Modeling
  - Analysis of Display Content
  - Analysis of the Work Environment
- Interface Design Steps
  - Applying Interface Design Steps
  - User Interface Design Patterns
  - Design Issues
- Web App Interface Design
  - Interface Design Principles and Guidelines Interface Design Workflow for Web Apps
- Design Evaluation

Introduction



Why Should You Care about Interface Design Principles?

- Interface inconsistency can cost a big company millions of dollars in lost productivity.
- The software becomes more popular if its user interface is:
  - Attractive
  - Simple to use
  - Responsive in short time
  - Clear to understand
  - Consistent on all interfacing screens

Background

- Interface design focuses on the following
  - The design of interfaces between software components
  - The design of interfaces between the software and other nonhuman producers and consumers of information
  - The design of the interface between a human and the computer
- Graphical user interfaces (GUIs) have helped to eliminate many of the most horrific interface problems
- However, some are still difficult to learn, hard to use.
- User interface analysis and design has to do with the study of people and how they relate to technology

The Golden Rules

- 1. Place the user in control.
- 2. Reduce the user’s memory load.
- 3. Make the interface consistent.

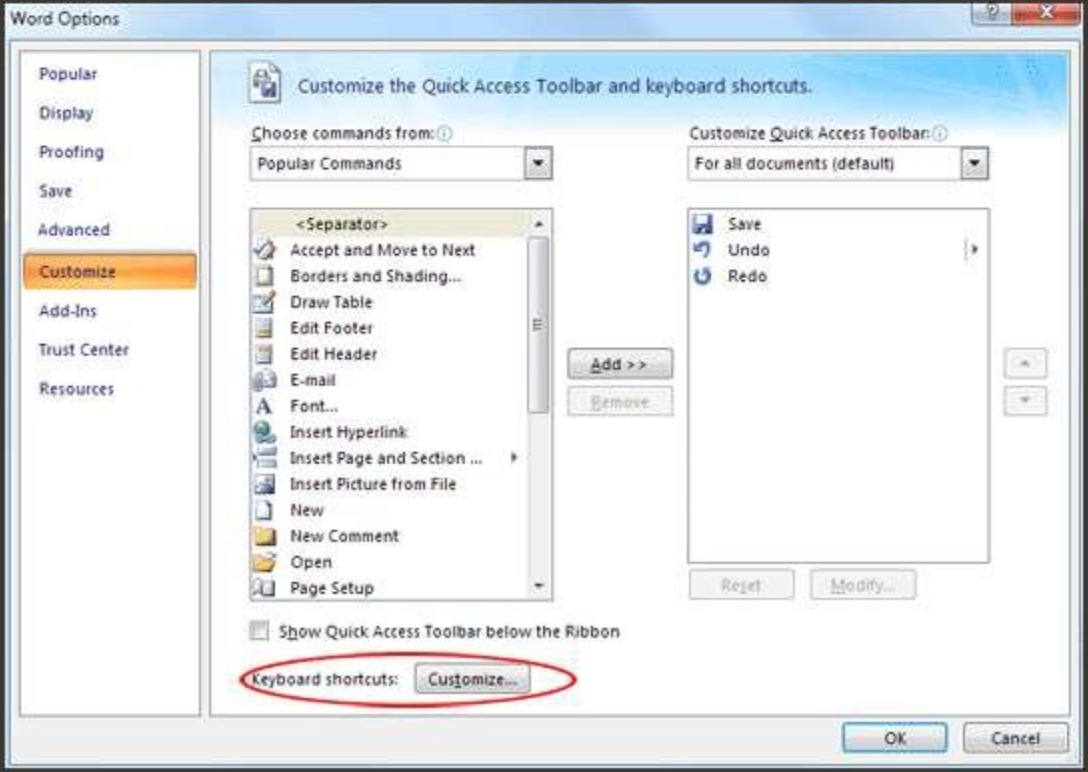
Place the user in control

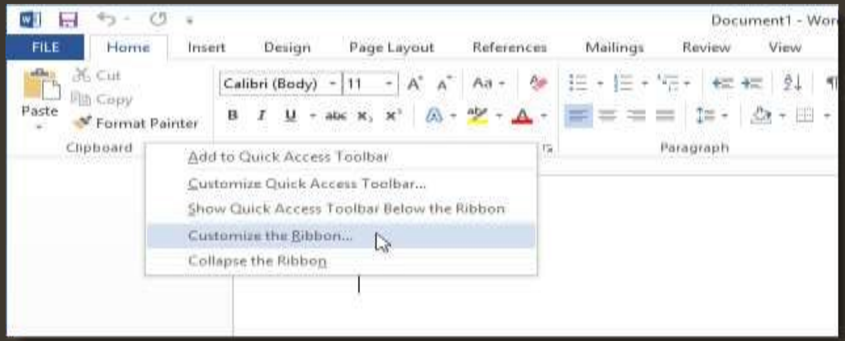
- ***Provide for flexible interaction***
  - ***Example***
    - Because different users have different interaction preferences, choices should be provided. For example, software might allow a user to interact via keyboard commands, mouse movement, a digitizer pen, a multitouch screen, or voice recognition commands
- ***Define interaction modes in a way that does not force a user into unnecessary or undesired actions.***
  - ***Example***
    - If spell check is selected in a word-processor menu, the software moves to a spell-checking mode. There is no reason to force the user to remain in spell-checking mode if the user desires to make a small text edit along the way. The user should be able to enter and exit the mode with little or no effort.

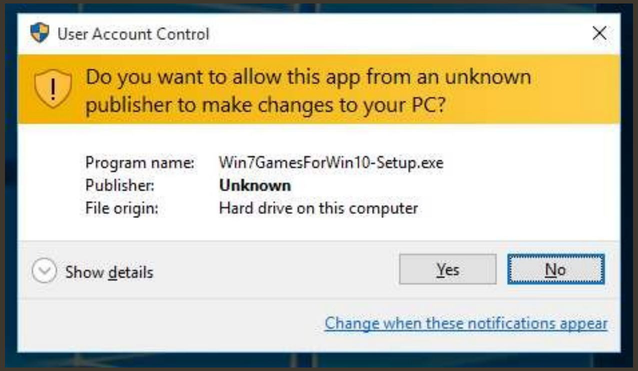
Place the user in control

- ***Allow user interaction to be interruptible and undoable***.
  - The user should also be able to “undo” any action.
- ***Hide technical internals from the casual user.***
  - The user should not be aware of the operating system, file management functions
- ***Design for direct interaction with objects that appear on the screen***
  - The user feels a sense of control when able to manipulate the objects that are necessary to perform a task in a manner similar to what would occur if the object were a physical thing.

Example

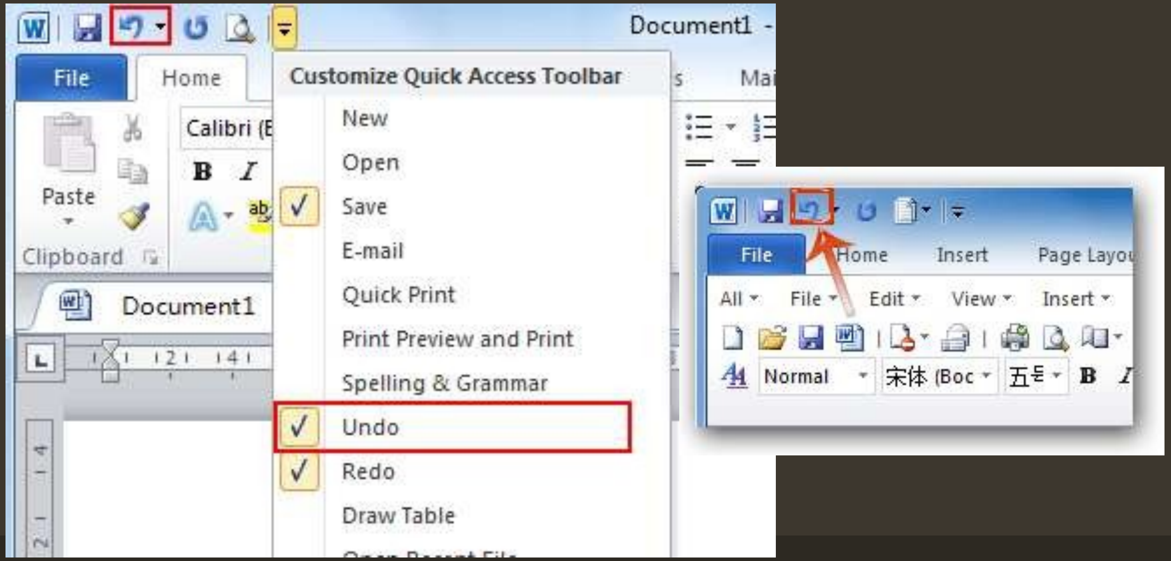





User Account Control

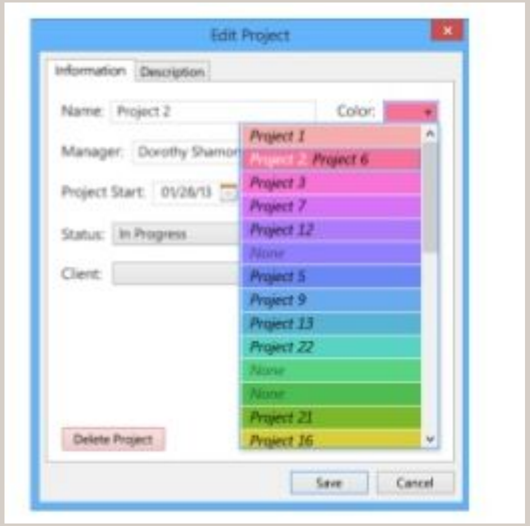
Do you want to allow this app from an unknown publisher to make changes to your PC?

Program name: Win7GamesForWin10-Setup.exe
Publisher: **Unknown**
File origin: Hard drive on this computer



Reduce the user’s memory load.

- Reduce demand on short-term memory.
  - The interface shall reduce the user's requirement to remember past actions and results by providing visual cues of such actions
- Establish meaningful defaults
  - The system shall provide the user with default values that make sense to the average user but allow the user to change these defaults


Edit Project

Information | Description

Name: Project 2   Color:

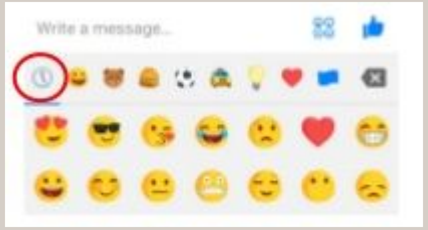
Manager: Dorothy Shamon

Project Start: 01/26/13

Status: In Progress

Client:

Project 1
Project 2 Project 6
Project 3
Project 7
Project 12
None
Project 5
Project 9
Project 13
Project 22
None
None
Project 21
Project 16

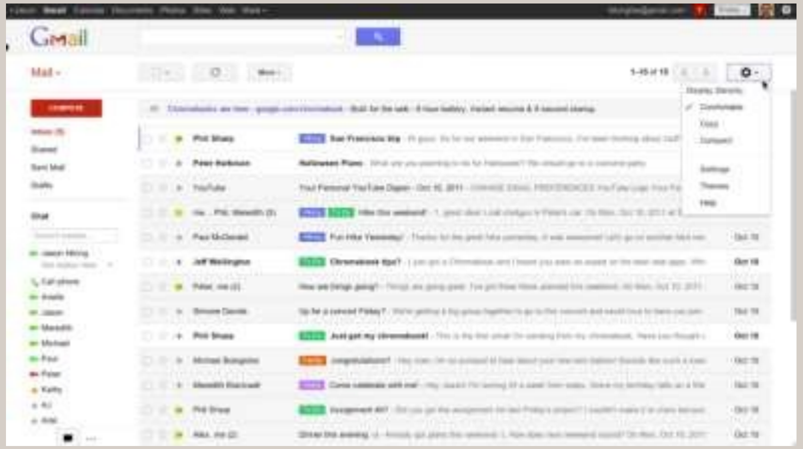


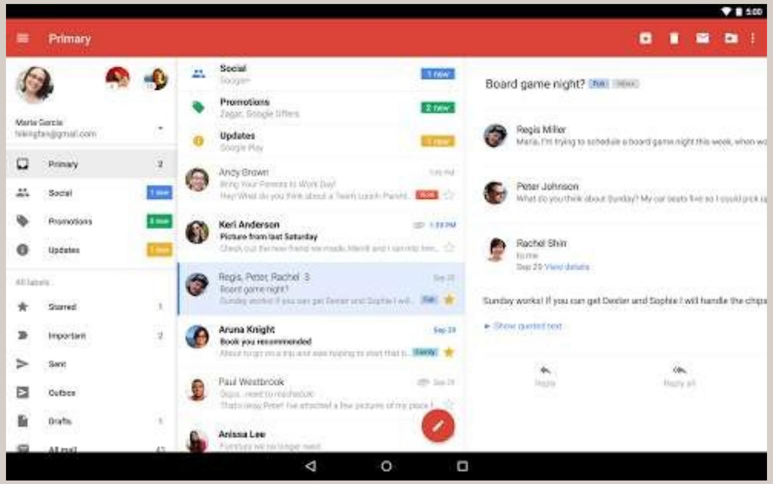
Reduce the user’s memory load cont.

- ***Define shortcuts that are intuitive***
  - The user shall be provided mnemonics (i.e., control or alt combinations)
- ***The visual layout of the interface should be based on a real-world metaphor***
  - The screen layout of the user interface shall contain well-understood visual cues that the user can relate to real-world actions
- ***Disclose information in a progressive fashion.***
  - When interacting with a task, an object or some behavior, the interface shall be organized hierarchically by moving the user progressively in a step-wise fashion from an abstract concept to a concrete action (e.g., text format options format dialog box.

Make the Interface Consistent

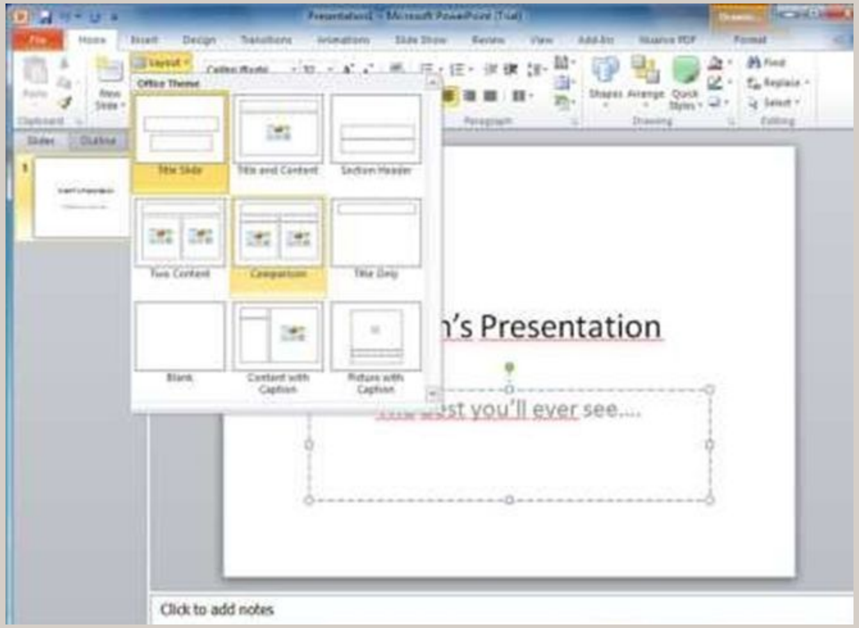
- ***Allow the user to put the current task into a meaningful context.***
  - All visual information shall be organized according to a design standard that is maintained throughout all screen displays
  - Input mechanisms shall be constrained to a limited set that is used consistently throughout the application
  - Mechanisms for navigating from task to task shall be consistently defined and implemented
  - .The interface shall provide indicators (e.g., window titles, consistent color coding) that enable the user to know the context of the work at hand





Make the Interface Consistent

- ***Maintain consistency across a family of applications***
  - A set of applications performing complimentary functionality shall all implement the same design rules so that consistency is maintained for all interaction



User Interface Analysis and Design Models

Concepts of Good/Bad Design

- **Affordances**
  - Perceived properties of an artifact that determines how it can be used (e.g knobs/buttons/slots)
- **Constraints**
  - Physical, semantic, cultural, and logical factors that encourage proper actions
- **Conceptual Models**
- Mental model of system which allows users to:
  - understand the system
  - predict the effects of actions
  - interpret results
- **Mappings**
  - Describe relationship between controls and their effects on system
- **Visibility**
  - The system shows you the conceptual model by showing its state and actions that can be taken
- **Feedback**
  - Information about effects of user's actions

User Interfaces Analysis & Design

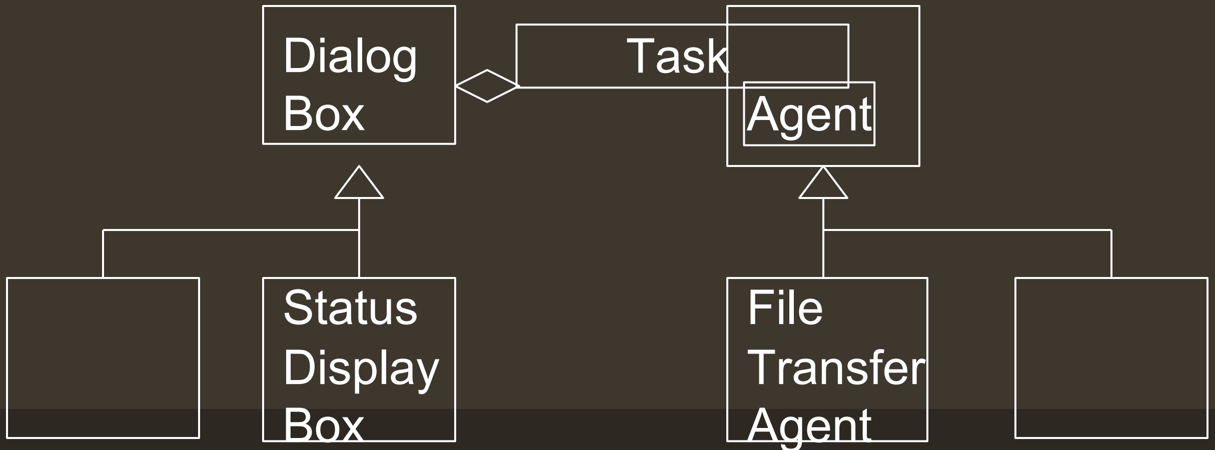
- ***Interface Analysis and Design model***
  - Four different models come into play when a user interface is to be analyzed and designed
    - **User Model.**
    - **Design Model,**
    - **User’s Mental Model**
    - **Implementation Model.**

1. User Profile Model

- Establishes the profile of the end-users of the system
  - Based on age, gender, physical abilities, education, cultural or ethnic background, motivation, goals, and Personality
- Considers syntactic knowledge of the user
  - The mechanics of interaction that are required to use the interface effectively
- Considers semantic knowledge of the user
  - The underlying sense of the application; an understanding of the functions that are performed, the meaning of input and output, and the objectives of the system

2. Design Model

- Derived from the analysis model of the requirements
- Incorporates data, architectural, interface, and procedural representations of the software
- Constrained by information in the requirements specification that helps define the user of the system
- Normally is incidental to other parts of the design model
  - But in many cases it is as important as the other parts

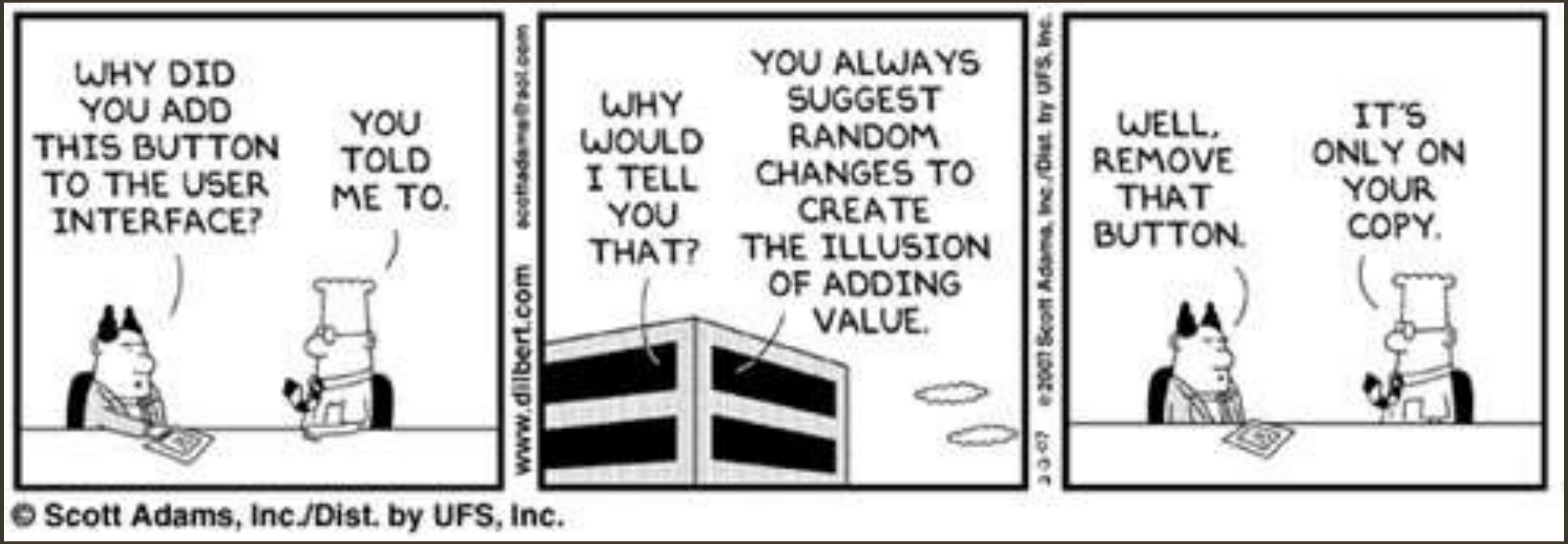


3.Implementation Model

- Consists of the look and feel of the interface combined with all supporting information (books, videos, help files) that describe system syntax and semantics
- Strives to agree with the user's mental model; users then feel comfortable with the software and use it effectively
- Serves as a translation of the design model by providing a realization of the information contained in the user profile model and the user’s mental model

4.User's Mental Model

- Often called the user's system perception
- Consists of the image of the system that users carry in their heads
- Accuracy of the description depends upon the user's profile and overall familiarity with the software in the application domain


WHY DID YOU ADD THIS BUTTON TO THE USER INTERFACE?
YOU TOLD ME TO.
WHY WOULD I TELL YOU THAT?
YOU ALWAYS SUGGEST RANDOM CHANGES TO CREATE THE ILLUSION OF ADDING VALUE.
WELL, REMOVE THAT BUTTON.
IT'S ONLY ON YOUR COPY.
www.dilbert.com
scottadams@aol.com
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
2-3-07
© Scott Adams, Inc./Dist. by UFS, Inc.

Process

- The analysis and design process for user interfaces is iterative and can be represented using a spiral model.
- (1) interface analysis and modeling
- (2) interface design
- (3) interface construction
- (4) interface validation.

Interface Design Process

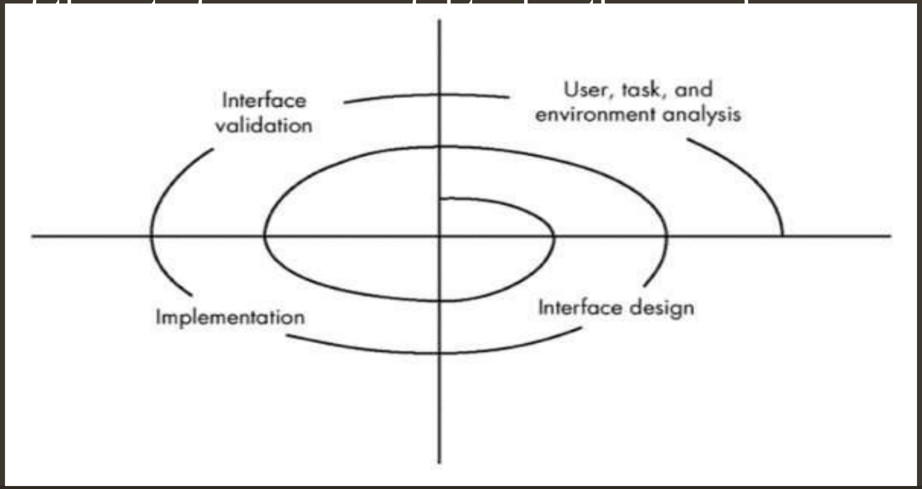
- User interface development follows a spiral process
  - Interface analysis (user, task, and environment analysis)
    - Focuses on the profile of the users who will interact with the system
    - Concentrates on users, tasks, content and work environment
    - Studies different models of system function (as perceived from the outside)
    - Delineates the human- and computer-oriented tasks that are required to achieve system function
  - Interface design
    - Defines a set of interface objects and actions (and their screen representations) that enable a user to perform all defined tasks in a manner that meets every usability goal defined for the system

Analysis of the user environment focuses on the physical work environment.

- Question
  - • Where will the interface be located physically?
  - • Will the user be sitting, standing, or performing other tasks unrelated to the interface?
  - • Does the interface hardware accommodate space, light, or noise constraints?
  - • Are there special human factors considerations driven by environmental factors?

Interface Design Process

- Interface construction
  - Begins with a prototype that enables usage scenarios to be evaluated
  - Continues with development tools to complete the construction
- Interface validation, focuses on
  - The ability of the interface to implement every user task correctly, to accommodate all task variations, and to achieve all general user requirements
  - The degree to which the interface is easy to use and easy to learn
  - The users' acceptance


Interface validation
User, task, and environment analysis
Implementation
Interface design