CS 5150 Software Engineering

Three Popular Architectural Styles

William Y. Arms
Examples

- Electricity utility customer billing (e.g., NYSEG)
- Telephone call recording and billing (e.g., Verizon)
- Car rental reservations (e.g., Hertz)
- Stock market brokerage (e.g., Charles Schwab)
- E-commerce (e.g., Amazon.com)
- University grade registration (e.g., Cornell)
Example: Electricity Utility Billing
Requirements analysis identifies several transaction types:

• Create account / close account
• Meter reading
• Payment received
• Other credits / debits
• Check cleared / check bounced
• Account query
• Correction of error
• etc., etc., etc.,
First Attempt

Each transaction is handled as it arrives.
Criticisms of First Attempt

Where is this first attempt weak?

- All activities are triggered by a transaction
- A bill is sent out for each transaction, even if there are several per day
- Bills are not sent out on a monthly cycle
- Awkward to answer customer queries
- No process for error checking and correction
- Inefficient in staff time
Batch Processing: Edit and Validation

- Batches of incoming transactions → Data input → Edit & validation
- Edit & validation → errors → Master file (read only)
- Batches of validated transactions
Deployment Diagram: Validation

DataInput

EditCheck

RawData

ValidData

MasterFile

Check
Batch Processing: Master File Update

Validated transactions in batches → Sort by account → Batches of input data → Master file update → Bills

Errors → Reports

Checkpoints and audit trail
Benefits of Batch Processing with Master File Update

- All transactions for an account are processed together at appropriate intervals, e.g., monthly
- Backup and recovery have fixed checkpoints
- Better management control of operations
- Efficient use of staff and hardware
- Error detection and correction is simplified
Advantages:
Efficient way to process batches of transactions.

Disadvantages:
Information in master file is not updated immediately. No good way to answer customer inquiries.

Example: billing system for electric utility
Online Inquiry

A customer calls the utility and speaks to a customer service representative.

Customer service department can read the master file, make annotations, and create transactions, but cannot change the master file.
The representative can read the master file, but not make changes to it.

If the representative wishes to change information in the master file, a new transaction is created as input to the master file update system.
Architectural Style: Master File Update (Full)

Advantage:
Efficient way to answer customer inquiries.

Disadvantage:
Information in master file is not updated immediately.

Example: billing system for electric utility
Real Time Transactions

Example: A small bank

- Transactions are received by customer in person, over the Internet, by mail or by telephone.
- Some transactions must be processed immediately (e.g., cash withdrawal), others are suitable for overnight processing (e.g., check clearing).
- Complex customer inquiries.
- Highly competitive market.
Real-time Transactions & Batch Processing

This is a combination of the Repository style and the Master File Update style.
Practical Consideration

- Can real-time service during scheduled hours be combined with batch processing overnight?
- How will the system guarantee database consistency after any type of failure?
  - reload from checkpoint + log detailed audit trail
- How will **transaction errors** be avoided and identified?
- How will **transaction errors** be **corrected**?
- How will **staff dishonesty** be controlled?

These practical considerations may be major factors in the choice of architecture.
Many data intensive systems, e.g., those used by banks, universities, etc. are legacy systems. They may have been developed forty years ago as batch processing, master file update systems and been continually modified.

- Recent modifications might include customer interfaces for the web, smartphones, etc.
- The systems will have migrated from computer to computer, across operating systems, to different database systems, etc.
- The organizations may have changed through mergers, etc.

Maintaining a coherent system architecture for such legacy systems is an enormous challenge, yet the complexity of building new systems is so great that it is rarely attempted.
The basic client/server architecture of the web has:

- a server that delivers static pages in HTML format
- a client (known as a browser) that renders HTML pages

Both server and client implement the HTTP interface.

**Problem**

Extend the architecture of the server so that it can configure HTML pages dynamically.
Web Server with Data Store

Advantage:
Server-side code can configure pages, access data, validate information, etc.

Disadvantage:
All interaction requires communication with server
Architectural Style: Three Tier Architecture

Each of the tiers can be replaced by other components that implement the same interfaces
These components might be located on a single node.
Three Tier Architecture: Broadcast Searching

This is an example of a **multicast** protocol.

The primary difficulty is to avoid troubles at one site degrading the entire system (e.g., every transaction cannot wait for a system to time out).
Extending the Architecture of the Web

Using a three tier architecture, the web has:

• a server that delivers dynamic pages in HTML format
• a client (known as a browser) that renders HTML pages

Both server and client implement the HTTP interface.

Problem 2

Every interaction with the user requires communication between the client and the server.

Extend the architecture so that simple user interactions do not need messages to be passed between the client and the server.
Extending the Web with Executable Code that can be Downloaded

Executable code in a scripting language such as JavaScript can be downloaded from the server

**Advantage:**
- Scripts can interact with user and process information locally

**Disadvantage:**
- All interactions are constrained by web protocols
In this example, each package represents a related set of classes.
Extending the Architecture of the Web

Using a three tier architecture with downloadable scripts, the web has:

• a server that delivers dynamic pages in HTML format
• a client (known as a browser) that renders HTML pages and executes scripts

Both server and client implement the HTTP interface.

Problem 3

Every interaction between the client and a server uses the HTTP protocol.

Extend the architecture so that other protocols can be used.
Web User Interface: Applet

- Any executable code can run on client
- Client can connect to any server
- Functions are constrained by capabilities of browser
Applet Interfaces

- XYZInterface
- WebBrowser
- HTTP
- XYZServer
- WebServer
Extending the Architecture of the Web

These examples (three tier architecture, downloadable scripts, and applets) are just some of the ways in which the basic architecture has been extended. Here are some more:

Protocols:
- HTTP, FTP, etc., proxies

Data types:
- helper applications, plug-ins, etc.

Executable code:
- Server-side code, e.g., servlets, CGI

Style sheets:
- CSS, etc.
System Design Study 3
Model/View/Controller for Mobile Apps

Diagram:
- **Model**: Central component, responsible for data and logic.
  - **State query**: Interaction with the Model to retrieve data.
  - **State change**: Interaction with the Model to update data.
- **View**:视觉展示层
  - **View control**: Interaction with the View to manage user interface.
- **Controller**: 控制层
  - **View control**: Interaction with the Controller to handle user inputs.
The model records the state of the application and notifies subscribers. It does not depend on the controller or the view.

• stores the state of the application in suitable data structures or databases
• responds to instructions to change the state information
• notifies subscribers of events that change the state
• may be responsible for validation of information
The **view** is the part of the user interface that presents the state of the interface to the user. It subscribes to the model, which notifies it of events that change the state.

- renders data from the model for the user interface
- provides editors for properties, such as text fields, etc.
- receives updates from the model
- sends user input to the controller

A given model may support a choice of alternative views.
The controller is the part of the user interface that manages user input and navigation within the application.

- defines the application behavior
- maps user actions to changes in the state of the model
- interacts with external services via APIs
- may be responsible for validation of information

Different frameworks handle controllers in different ways. In particular there are several ways to divide responsibilities between the model and the controller, e.g., data validation, external APIs.
Mobile apps often make extensive use of cloud-based external services, each with an API (e.g., location, validation). These are usually managed by the controller.
Example: Control of a unmanned model aircraft

**Controller:** Receives instrument readings from the aircraft and sends controls signals to the aircraft.

**Model:** Translates data received from and sent to the aircraft, and instructions from the user into a model of flight performance. Uses domain knowledge about the aircraft and flight.

**View:** Displays information about the aircraft to the user on the ground and transmits instructions to the model.
There are many variants of the common architectural styles. Do not be surprised if you encounter a variant that is different from the one described in this course.

This is particularly true with the Model-View-Controller style. Several programming frameworks call classes that implement a variant of the Model-View-Controller architectural style a design pattern.

In this course we distinguish carefully between architectural styles and design patterns.

Architectural styles are part of system design. They are defined in terms of subsystems, components, and deployment.

Design patterns are part of program design. They are defined in terms of classes.