

# CS211 Lecture: Architectural Design; Packages; Layered Systems

October 18, 2007

## *Objectives:*

1. To introduce architectural design
2. To introduce UML diagrams relevant to architectural design - component diagram, deployment diagram
3. To discuss additional architectural patterns (in addition to MVC): Layered, Client-Server, Pipe and Filter, Transaction Processing

## *Materials:*

1. Projectable of example component diagram
2. Projectable of Java layered architecture
3. Projectable of alternative client-server architectures
4. Projectable of pipe-and-filter architecture
5. Projectable of pipe-and-filter structure of a compiler
6. Projectable of transaction-processing architecture

## **I. Introduction**

A. There are many kinds of design that go into developing a software system.

Examples?

ASK

1. Architectural design
2. Class design
3. User interface design
4. Database design

We talked about detailed design of classes recently, and we will talk about user interface design and database design later in the course. Today we focus on architectural design.

B. The highest level of design is sometimes called architectural design.

1. We can speak of system architecture, which deals with the overall structure of the system in terms of its subsystems and major components.

For example, consider the anti-lock brake system of a car. The purpose of such a system is to prevent hard braking from locking the wheels, which produces skidding.

Such a system consists of speed sensors for each wheel, hydraulic valves that can be used to reduce braking pressure when one or more wheels lock up, and a software controller that senses when one or more wheels is rotating considerably slower than the others, indicating that it is about to lock up. In such a situation, the ABS controller signals the hydraulic valves to reduce braking pressure, thus allowing the wheel that is about to lock up to turn again.

The major components of this system, then, are 4 wheel sensors, some number of hydraulic valves, a microprocessor and software that runs on the microprocessor.

2. We can also speak of the architecture of an individual software component.
- C. In any sort of design, one key thing we need to do is to partition a larger problem into smaller pieces. (The divide and conquer principle). At the system architectural level, the pieces go by different names, depending on what is actually involved:
1. A component is a piece of an overall system that has a clear role, can be isolated, and can, in principle, be replaced with a different component that provides the same functionality. A component may be software, hardware, or both.

Examples:

- a) A hardware device like a computer's monitor is a component.
- b) A web-browser is a component.
- c) In designing software, a key concept is the notion of a re-usable component, which is simply a component that can be "plugged in" to different problems

Thus, in our ABS example, the components are the wheel speed sensors, the hydraulic valves, and the software running on the ABS Controller system.

2. Sometimes the term module is used for a component that is implemented in a programming language

Thus, in our ABS example, the software running on the ABS Controller system is a module.

3. A system is a collection of components that work together to do some job. Complex systems are often composed of definable subsystems, which have internal structure of their own.

(Sometimes the term “system” is limited to just hardware and software, but other times the term is used more broadly to encompass people, processes, organizational structures etc.)

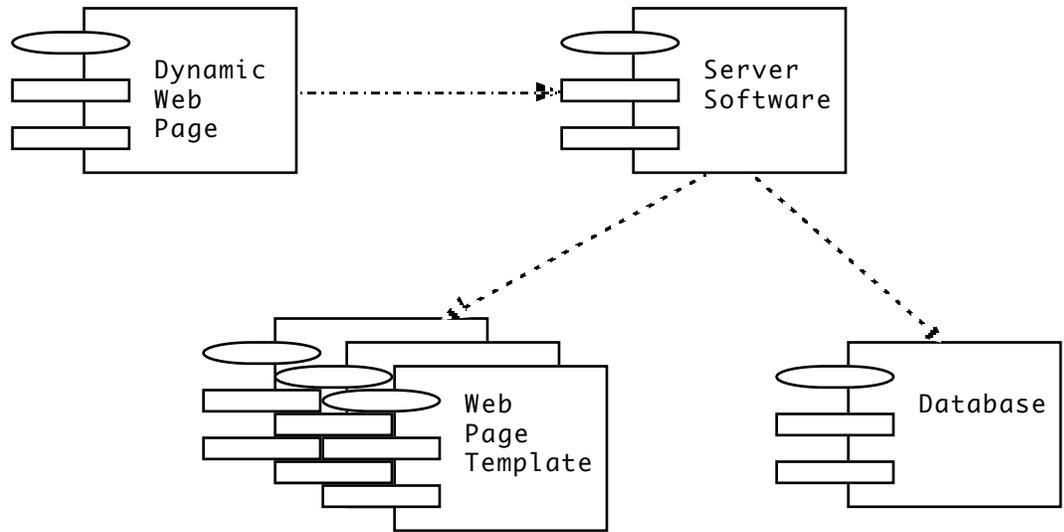
In our ABS Controller example, it is likely the case that the wheel sensors and hydraulic valves are subsystems which might themselves be further analyzed.

4. When a system is geographically distributed, the subsystem at a particular location is often called a node.

D. UML has two kinds of diagrams that are useful at this level.

1. A component diagram shows the major software components of a system and their relationships.
  - a) Figure 9.7 in the book is an example of a component diagram.
  - b) Another example: A typical structure for an e-commerce site is as follows: the web server houses a database of information about the company’s products. When a customer request is received, it generates a dynamic web page which it sends to the customer’s browser, based on a template stored on the server. (E.g. if the customer submits a search request, the server software searches the database and creates an appropriate search results page based on a template for such pages.) This can be depicted as follows (using UML 1 notation, which is the notation used in the book - UML 2 is different)

PROJECT



2. A deployment diagram shows the major hardware components of a system and their relationships. It is particularly useful for use with distributed systems, in which the hardware components are located at different places.

- a) Figure 9.8 in the book is an example of a deployment diagram
- b) We will look at these more fully when we consider distributed systems later in the course.

## II. Architectural Patterns

- A. It turns out that there are a variety of architectural design patterns that are appropriate for different situations.
- B. We have talked about one extensively, which is primarily useful for analyzing the architecture of software components.

ASK

Model view controller

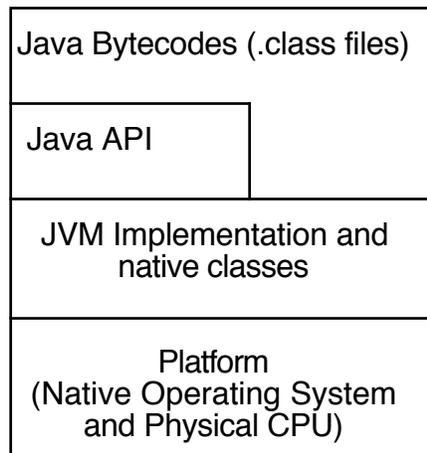
- C. The book talked about another such pattern, which can be used both for hardware/software systems and for understanding purely software systems.

ASK

## Layered architecture

1. In a system using a layered architecture, the system is organized into layers.
  - a) A given layer may only directly depend on the services provided by the layer immediately below it.
  - b) Each layer defines an interface representing services it makes available to the next layer up.
2. Example: Java implementations typically use a layered architecture that looks like this:

### PROJECT



- a) A JVM implementation is specific to a particular platform. It depends on the facilities of the native operating system, and is capable of executing any standard java class file.
  - b) The bytecodes produced by the javac compiler (and the API routines in the java. and javax. packages in the system library) depend only on the standard java class file format. Thus, the same class files will run on any java implementation, regardless of the platform.
3. Layered architectures are not confined to software. Layering is also used in systems comprised of software and hardware - e.g. there is a model for understanding computer networks called the ISO/OSI model which has seven distinct layers - several of which involve both software and hardware.

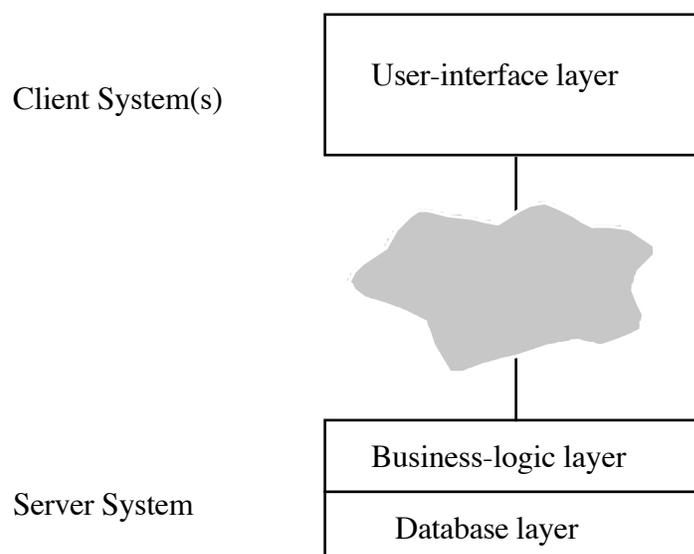
The lowest layer is the physical layer- the physical means by which bits are transmitted from one place to another. A physical link can be something like ethernet, or fiber optics, or wireless. Higher layers should be unaffected by what choice is made at this layer, though. One manifestation of this is that a computer that is connected to a network via a wired connection can be seamlessly shifted to a wireless connection.

D. Another pattern is the client-server architecture - a pattern that occurs over and over in Internet applications.

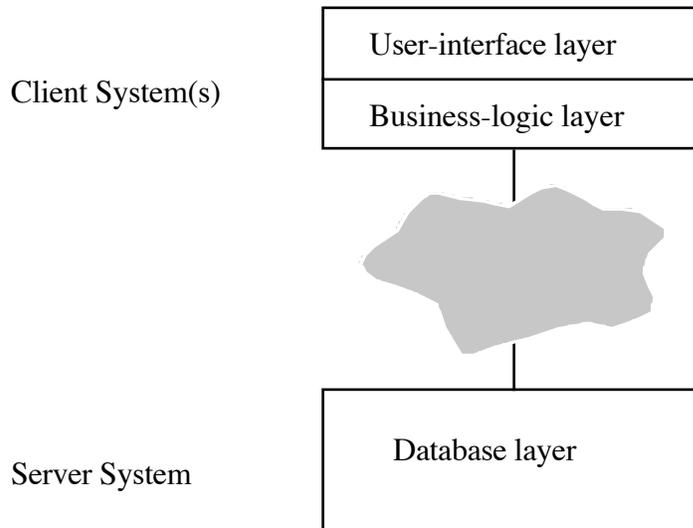
1. In the simplest case, a client-server system consists of a server system and (one or more) client subsystems. For example, a web browser relates as a client to a web server; the mail program running on a personal computer acts as client to a mail server, etc.
2. More complex systems can be understood in terms of a layered model: a user-interface layer, a business-logic layer, and a database layer. For example, many e-commerce systems are set up this way: the user interface layer is a web page (perhaps with embedded javascript) viewed by a web browser; the business-logic layer is the software that provides information in response to user requests and processes orders, and the database layer stores information about the products and records user orders. There are three different ways these might be distributed:

#### PROJECT

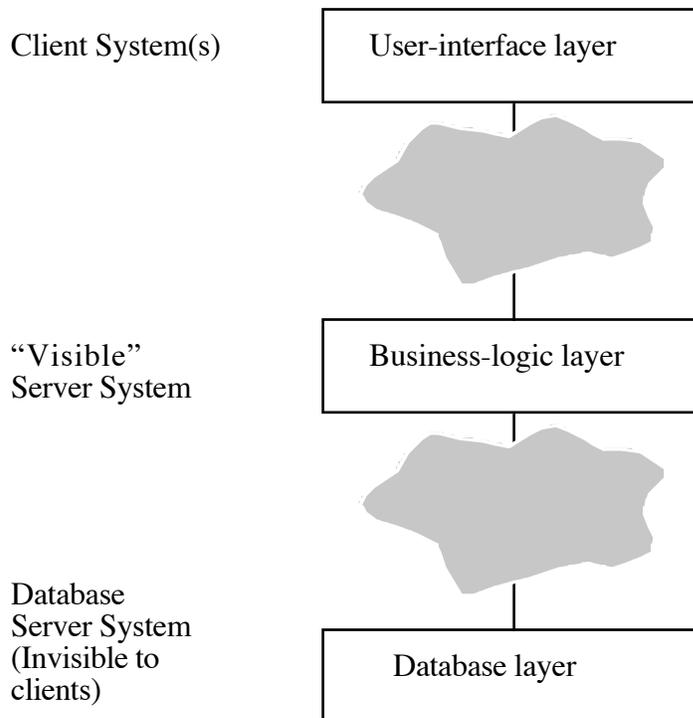
- a) An approach often used by e-commerce systems (the so-called “thin client” approach)



- b) The so-called “thick client” approach, used when it is desirable to install the business logic software on the client system (This wouldn’t work for e-commerce, of course, but is sometimes used for specialized applications)



- c) A three-layer approach that can also be used as an alternative to the thin client approach. (Note that the client would never see any difference between this approach and the thin-client approach; in fact, many e-commerce systems are in fact built this way)



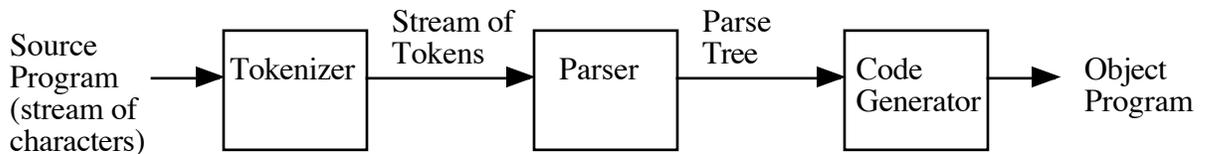
3. While the client-server architecture is most commonly seen in distributed systems, it can also be used for software systems running on a single computer - e.g. a program that uses a relational database is often structured as a client relating to a separate database server program running on the same computer. (In fact, a database server program running on a computer may simultaneously be serving several different clients on that same machine.)
- E. An alternative architecture for distributed systems is a Peer-to-Peer architecture in which there is no designated server as such. Instead, each participating system can function either as a client, or as a server to some other system.
- F. Another architectural pattern is the “Pipe and Filter” Pattern. In this approach, a system is organized as a collection of subsystems called filters. The overall input to the system flows into the first filter, which performs some transformation on it and sends the transformed data to the second filter, which performs a further transformation on it and sends it to the third filter ...

PROJECT



1. One place where this architecture is often used is compilers. Thus, a simple compiler might be organized like this:

PROJECT

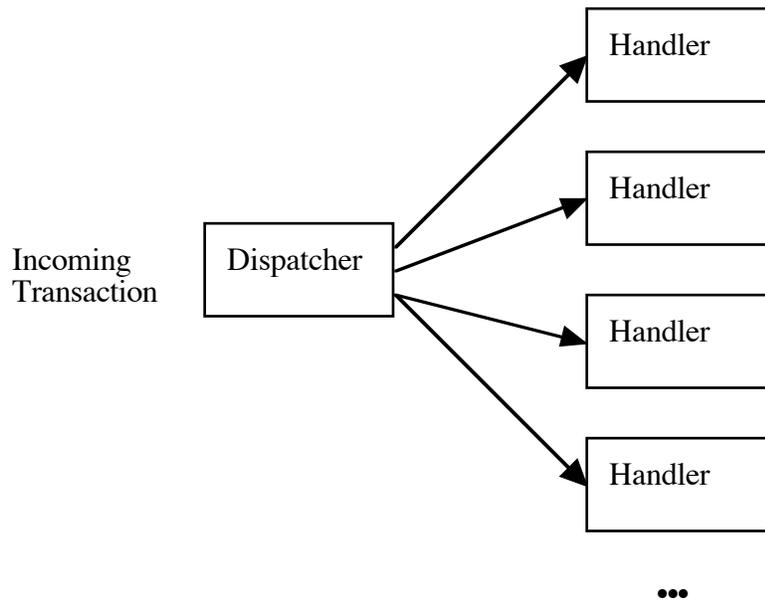


(More sophisticated compilers include various code-improvement (so-called “optimization”) steps which may be inserted either between the parser and the code generator, or after the code generator - or often both)

2. Another place where pipe and filter architecture is often used is in systems that process signals such as sounds (e.g. speech recognition systems.) In fact, the term “filter” really comes from the world of hardware signal-processing systems.

G. The Transaction-processing architecture. This is appropriate when the system basically exists to process discrete transactions of various types.

### 1. PROJECT



2. Note that this pattern is used in the ATM system, though the transactions are handled in the context of a session which the system also manages.