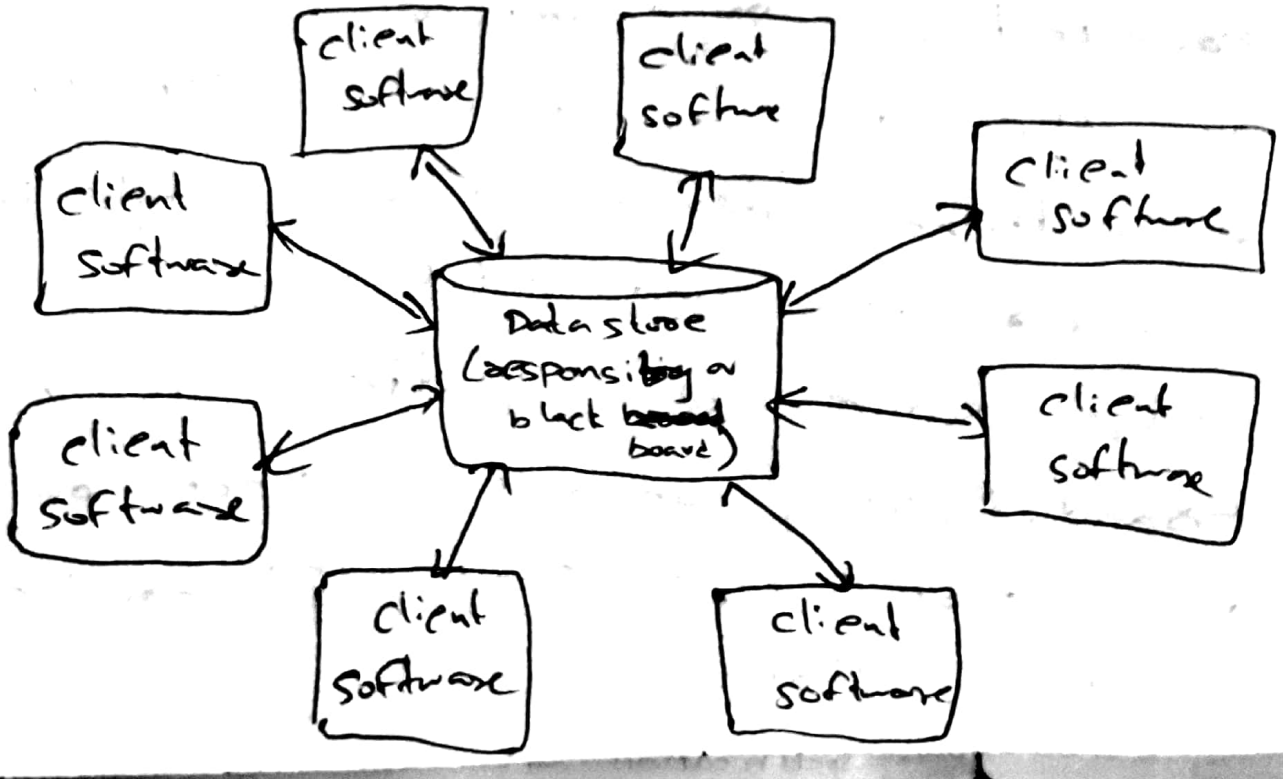
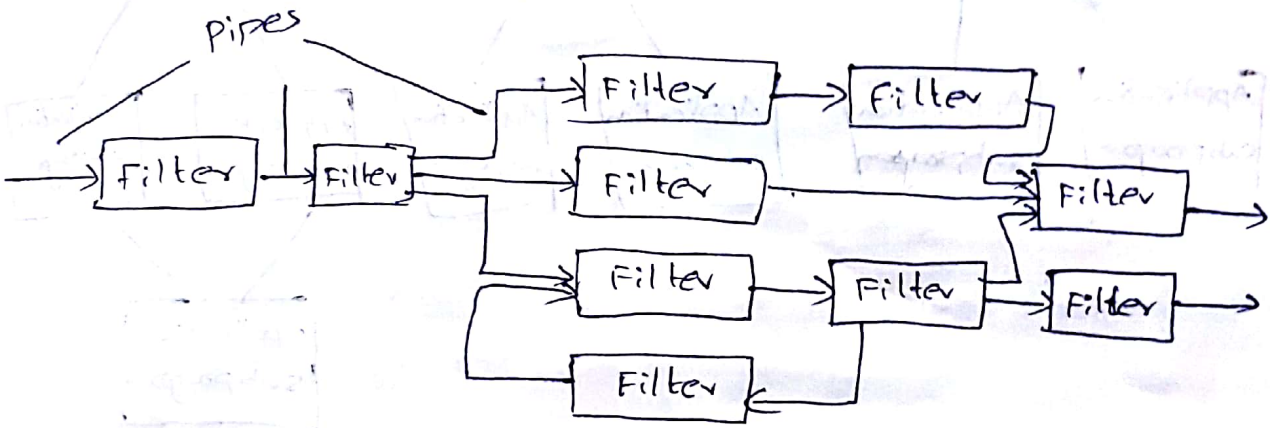
Datacentered Architecture

- A data store resides at the center of this architecture and is accessed frequently by other components that update, add, delete, or otherwise modify data within the store.
- This architecture promotes integrability. Existing components can be changed and new client components can be added to the architecture without concern about other clients.

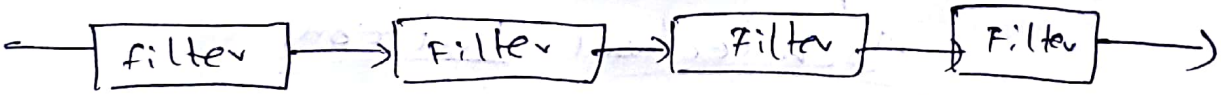


Data flow Architecture

- This architecture is applied when input data are to be transformed through a series of computational or manipulative components into output data.
- A pipe and filter structure has a set of components, called filters, connected by pipes that transmit data from one component to the next.



a) pipes and filter



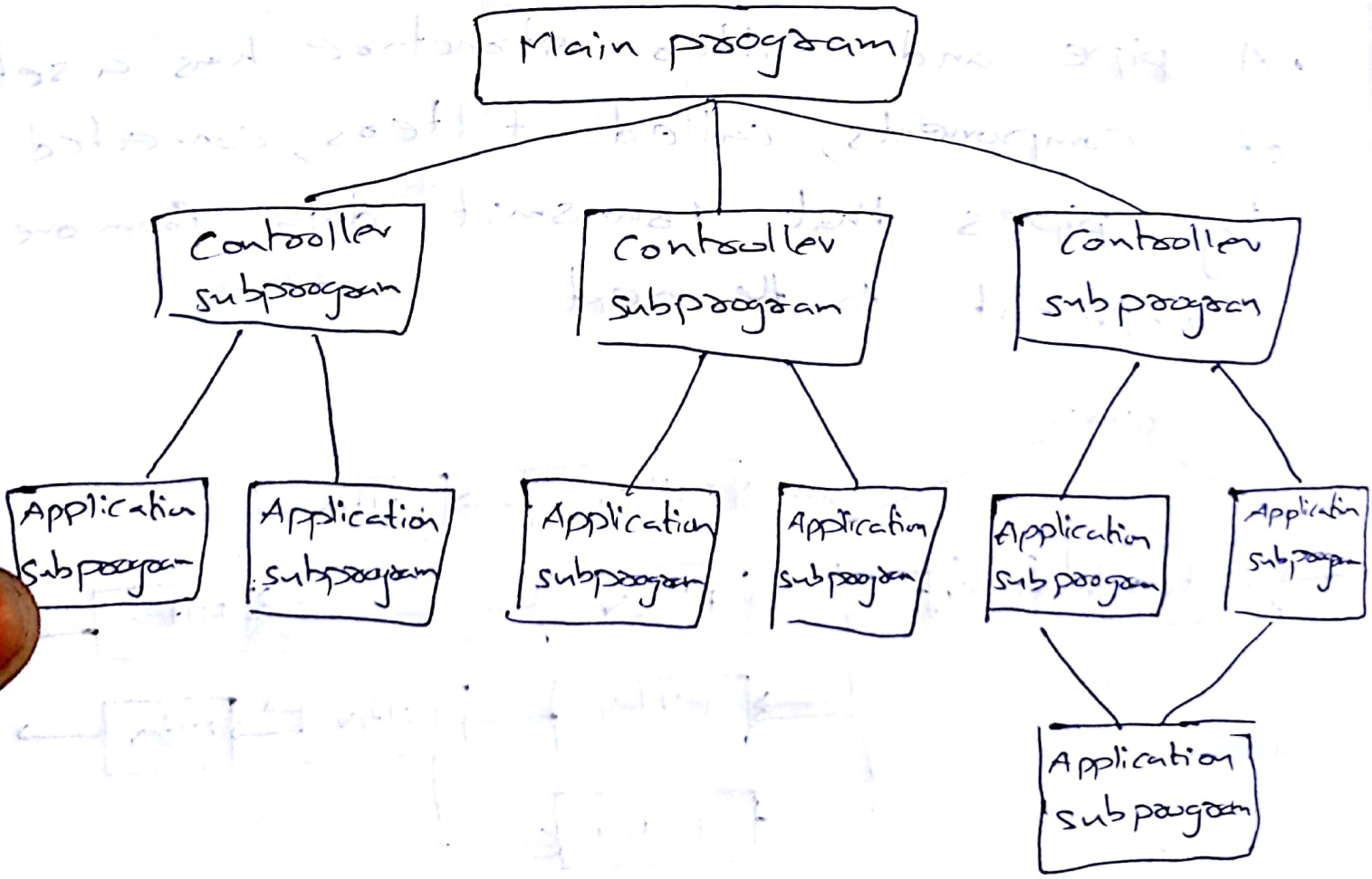
b) batch sequential.

Call and Return architecture

- The architectural style enables a S/w ~~describ~~ designer to acheive a program structure that is relatively easy to modify and scale.

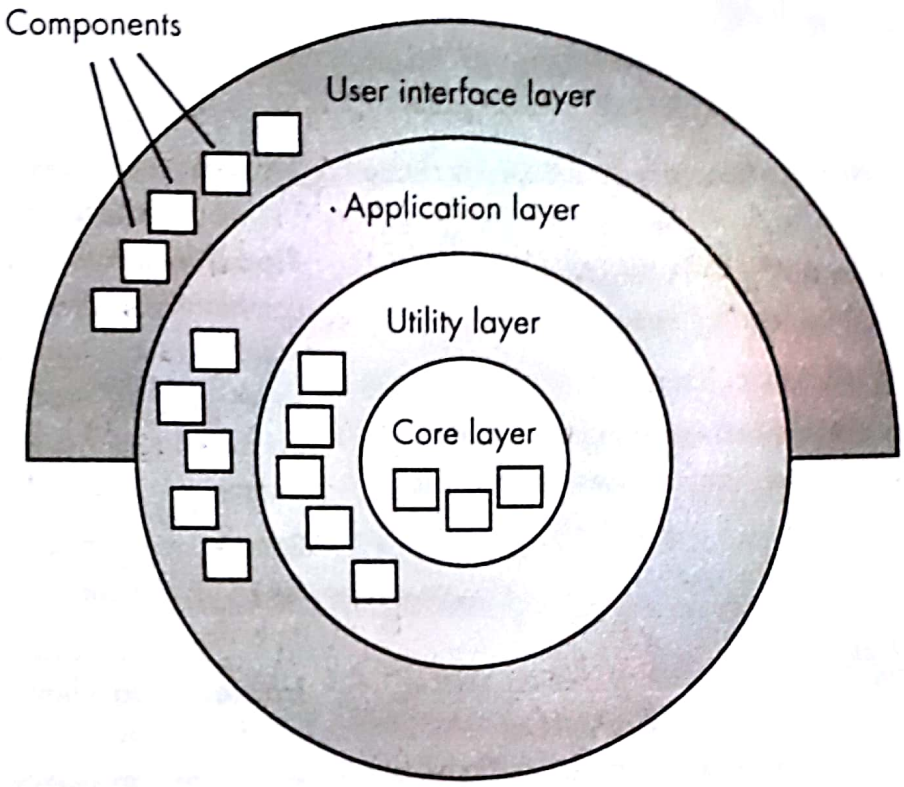
→ Main program/subprogram architecture:- The program structure is decomposed into a control hierarchy.

→ Remote procedure-call architecture: The components of the main program / subprogram are distributed across multiple computers on a network.



Object-oriented architectures. The components of a system encapsulate data and the operations that must be applied to manipulate the data. Communication and coordination between components are accomplished via message passing.

FIGURE 9.4
ayered
chitecture



Layered architectures. The basic structure of a layered architecture is illustrated in Figure 9.4. A number of different layers are defined, each accomplishing operations that progressively become closer to the machine instruction set. At the outer layer, components service user interface operations. At the inner layer, components perform operating system interfacing. Intermediate layers provide utility services and application software functions.

These architectural styles are only a small subset of those available.[2] Once requirements engineering uncovers the characteristics and constraints of the system to be built, the architectural style and/or combination of patterns that best fits those characteristics and constraints can be chosen. In many cases, more than one pattern might be appropriate and alternative architectural styles can be designed and evaluated. For example, a layered style (appropriate for most systems) can be combined with a data-centered architecture in many database applications.

Quote:

architectural patterns that address issues such as concurrency, persistence, and distribution.

Concurrency. Many applications must handle multiple tasks in a manner that simulates parallelism (i.e., this occurs whenever multiple "parallel" tasks or components are managed by a single processor). There are a number of different ways in which an

---

4 This implies that there will be a central foyer and hallway, that rooms will be placed to the left and right of the foyer, that the house will have two (or more) stories, that the bedrooms of the house will be upstairs, and so on. These "rules" are imposed once the decision is made to use the *center-hall colonial* style.

5 It is important to note that there is not universal agreement on this terminology. Some people (e.g., [BUS96]) use the terms *styles* and *patterns* synonymously, while others make the subtle distinction suggested in this section.

application can handle concurrency, and each can be presented by a different architectural pattern. For example, one approach is to use an *operating system process management* pattern that provides built-in OS features that allow components to execute concurrently. The pattern also incorporates OS functionality that manages communication between processes, scheduling, and other capabilities required to achieve concurrency. Another approach might be to define a task scheduler at the application level. A *task scheduler* pattern contains a set of active objects that each contains a *tick()* operation [BOS00]. The scheduler periodically invokes *tick()* for each object, which then performs the functions it must perform before returning control back to the scheduler, which then invokes the *tick()* operation for the next concurrent object.

Persistence. Data persists if it survives past the execution of the process that created it. Persistent data are stored in a database or file and may be read or modified by other processes at a later time. In object-oriented environments, the idea of a persistent object extends the persistence concept a bit further. The values of all of the object's attributes, the general state of the object, and other supplementary information are stored for future retrieval and use. In general, two architectural patterns are used to achieve persistence—a *database management system* pattern that applies the storage and retrieval capability of a DBMS to the application architecture or an *application level persistence* pattern that builds persistence features into the application architecture (e.g., word processing software that manages its own document structure).

Distribution. The distribution problem addresses the manner in which systems or components within systems communicate with one another in a distributed environment. There are two elements to this problem: (1) the way in which entities connect to one another, and (2) the nature of the communication that occurs. The most common architectural pattern established to address the distribution problem is the *broker* pattern. A broker acts as a "middle-man" between the client component and a server component. The client sends a message to the broker (containing all appropriate information for the communication to be effected), and the broker completes the connection. CORBA (Chapter 30) is an example of a broker architecture.

Before any one of the architectural patterns noted in the preceding paragraphs can be chosen, it must be assessed for its appropriateness for the application and the overall architectural style, as well as its maintainability, reliability, security, and performance.

10.3.3 Organization and Refinement

Because the design process often leaves a software engineer with a number of architectural alternatives, it is important to establish a set of design criteria that can be used to assess an architectural design. The following questions [BAS03] provide insight into the architectural style that has been derived.