Question 1: List two examples each of using pipe-and-filter styles and Batch Sequential style.

Pipe-and-filter example:

Consider a Web service for printing insurance policies. The service accepts XML messages from agency management systems. Incoming messages are based on the ACORD XML specification, an insurance industry standard. However, each agency has added proprietary extensions to the standard ACORD transactions. A print request message specifies the type of document to be generated, for example, an HTML document or a Portable Document Format (PDF) document. The request also includes policy data such as client information, coverage, and endorsements. The Web service processes the proprietary extensions and adds the jurisdiction-specific information that should appear on the printed documents, such as local or regional requirements and restrictions. The Web service then generates the documents in the requested format and returns them to the agency management system.
You could implement these processing steps as a single transformation within the Web service. Although viable, this solution does not let you reuse the transformation in a different context. In addition, to accommodate new requirements, you would have to change several components of the Web service. For example, you would have to change several components if a new requirement calls for decrypting some elements of the incoming messages.
An implementation that is based on Pipes and Filters provides an elegant alternative for the printing Web service. Figure 1 illustrates a solution that involves three separate transformations. The transformations are implemented as filters that handle conversion, enrichment, and rendering.

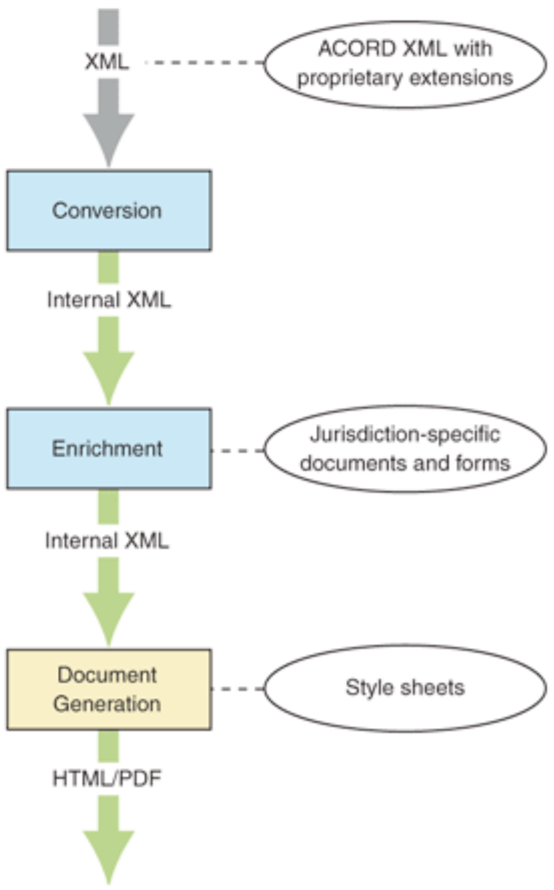

Figure 1. Printing Web service that uses Pipes and Filters

The printing service first converts the incoming messages into an internal vendor-independent format. This first transformation lowers the dependencies on the proprietary ACORD XML extensions. In effect, changing the format of the incoming messages only affects the conversion filter.

After conversion, the printing service retrieves documents and forms that depend on the jurisdiction and adds them to the request message. This transformation encapsulates the jurisdiction-specific enrichment.

When the message contains all the information that comprises the final electronic document, a document generation filter converts the message to HTML or PDF format. A style sheet repository provides information about the appearance of each document. This last transformation encapsulates the knowledge of rendering legally binding documents.

In this example, the Pipes and Filters implementation of the printing Web service has the following benefits that make it preferable to implementing the Web service as a single monolithic transformation:

Separation of concerns. Each filter solves a different problem.

Division of labor. ACORD XML experts implement the conversion of the proprietary extensions into an internal vendor-independent format. People who specialize in dealing with the intricacies of each jurisdiction assist with the implementation of the filter that handles those aspects. Formatters and layout experts implement document generation.

Specialization. Document-rendering is CPU intensive and, in the case of a PDF document, uses floating point operations. You can deploy the rendering to hardware that meets these requirements.

Reuse. Each filter encapsulates fewer context-specific assumptions. For example, the document generator takes messages that conform to some schema and generates an HTML or PDF document. Other applications can reuse this filter.

Batch Sequential example:

Figure 2 depicts an example of a transaction processing system that illustrates batch sequential processing style, where data passed from one step to another is stored on tapes.

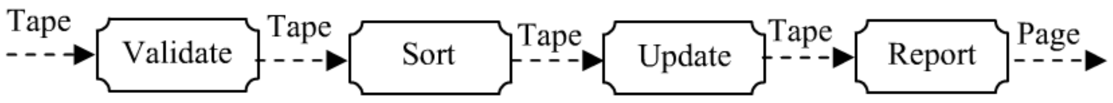

Figure 2. An example that illustrates batch sequential processing style

The system contains four components. The Validate component validates the validity of each transaction stored on a tape. It removes invalid transactions and stores all valid transactions on another tape. The Sort component executes when validation is completed. It reads in all transactions stored on the tape that the previous step generated and sorts the transactions according to the time that transactions are issued. The result is then stored in another tape. Since this step is after the completion of validation, all transactions stored on the tape must be valid and they must also be listed in the order that earlier transactions are listed first. After sorting the transactions, the Update component updates the contents of a business database by processing the transactions one by one according to their order stored on the tape generated by the Sort component. The results of the update are recorded on another tape, which was input to the final step of processing to generate a summary report by the Report component.

Question 2: Consider the “windows BAT program” and “Signal processing”. Which one is the Batch Sequential style and which one is the pipe & filter style. Give the reasons for your decision.

“Windows BAT program” is pipe-and-filter style whereas “Signal processing” is Batch Sequential style.

Reasons:
The “Windows BAT program” transforms all the data for each step. It runs completed data and next step begins while its processing starts just after the results produced. Therefore it is pipe-and-filter.
The “Signal processing” incrementally transforms the data and the processing mechanism is to wait for all the results in one process.

Question 3: Distinguish the difference between repository and black board architecture?

The means of communication distinguishes repository and blackboard.
Repository: a client sends a request to the system to perform a necessary action (e.g. insert data).
Blackboard: the system sends notification and data to subscribers when data of interest changes, and is thus active.