Chapter 5: Designing the Architecture

1. The NIST/ECMA model is closest to the layered architectural style because it is organized into parts where each part (payer) communicates principally with the parts on either side of it.

2.

   Pipe and filter: compilers, with the filters being the lexical analysis, parsing, semantic analysis and code generation processes.

   Client-server: The internet, with websites being hosted on web servers and browsers (clients) being used to access the sites. E-mail systems, with the mail clients communicating with the POP/IMAP mail servers. Multiplayer on-line games with the games being hosted on a central server and the players (clients) connecting to it.

   Peer-to-Peer: VoIP and instant messaging applications, where the peers communicate directly, without the need for a communication channel managed by a central server.

   Publish-subscribe: Graphical user interface-based applications, where different parts of the system provide functionality through notification of user interface events such as clicking a mouse button or placing the mouse cursor over a determined area of widget.

   Repositories: Business intelligence applications, where huge amounts of business and operation-related data are stored in a data warehouse. This data is then made available to be accessed by different business units for processing and analysis.

   Layered: Modern web browsers, where the HTML rendering engine, network protocol handling and operating system interface are the different layers in the architecture.

3. The repository design has high coupling because of the shared data, and has high cohesion since each module is responsible for one function of the system. The data abstraction design has lower coupling among components than the repository due to the use of interfaces to retrieve the necessary data. It has lower cohesion than the repository design since the primary functions are spread among the modules. The implicit invocation design has lower coupling than the previous designs due to the use of ADTs, and high cohesion as it separates the primary functions into separate modules. The pipe and filter design has the low coupling because the dependence between the filters is the pipe between them and has high cohesion because each independent filter relates to a primary function of the system.

4. Prototyping processing intensive applications does not result in saving any significant amount of development time. There is no way to create a prototype for such systems without first fully implementing the necessary operations and processes. Take a mathematics equation solving application as an example. Without the equation parsing and solving algorithms, the prototype would be used only to portray the user interface and would not result in any significant change in development time.

5. Systems with high user interaction benefit from prototyping by providing feedback on usability issues. Systems with complex protocols or transactions also benefit from prototyping by testing the implementation of such protocols and transactions.

6. Application generators are, conceptually, an advanced form of reuse. Modularity facilitates reuse by breaking functionality into pieces that can be manipulated and tailored individually in new contexts. This is even truer for application generation because the application-specific information used by a generator must be organized in small pieces so that it can be applied to one piece of functionality at a time.

7. In a shared data architecture, the components of the system manipulate the data directly and, as a result, are highly coupled to the data structure. In order to reuse the components of said system, it would require either having the same data model as the component or changing the component to operate with a new data model. Having the same data model is not always a viable option, as even though the systems may be similar the data may be completely different. Modifying the component to work with a new data model would probably take as much effort as writing a new component with the desired functionality.

8. The characteristics should include:
   a) cohesion
   b) coupling
   c) error, exception, and fault handling
   d) reusability
   e) modifiability
   f) clarity
   g) modularity

   For an operating system, probably the most important criteria would be error, exception and fault handling. Modifiability would also be important as operating systems tend to be used and maintained over long periods of time. Error, exception and fault handling would also be important for a word processor because of the user interactions. Reusability might also be important if a product line of word processors is envisioned. The same is true for a satellite tracking system, as such a system for a different satellite could reuse much of the same code. Cohesion, coupling, clarity and modifiablity are always important because they contribute to the ease of implementation.

9. Answers to this question may vary.

10. Questions (c), (d) and (e) deal with architectural design decisions. The types of report will affect how the reporting module should be designed as well as how it interacts with other modules. Both questions (d) and (e) deal with support for concurrency which should be addressed at the architectural level such as "will the application need to support more than one user at a time?", "will there be more than one instance of the application running at a time?" Question (a) deals with platform specific details such as computer architectures and should not be dealt with during the architectural design. Question (b) deals with the look and feel of the application and user interaction, and has no influence on architectural decisions. Similarly, question (f) deals with details of the depreciation algorithm implementation and do not contribute to the architectural design.

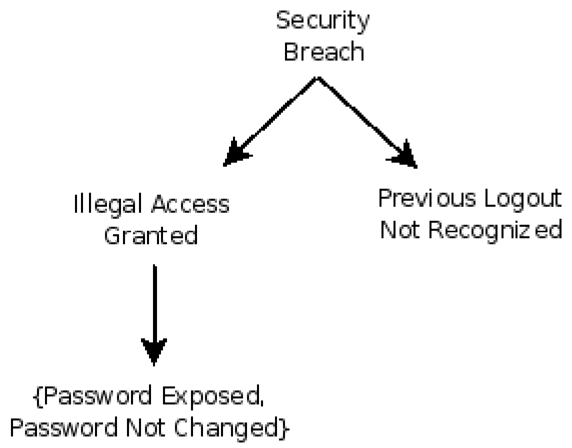
11.

   a) Use a repository software architecture. The central data store (bank server) contains the information about all the accounts in the bank. Each automated banking machine is a data accessor, able to retrieve account information for a given account and either withdraw or deposit cash into the account.
   b) Use a publish-subscribe software architecture. Each user is able to subscribe to the desired news bulletins, and will receive a notification from the news feeder whenever a new story is posted.
   c) Use a pipe-and-filter software architecture. Each different operation refers to a different filter (rotation, color tinting, cropping, etc) that can be combined ad-hoc to process the picture as desired.
   d) Use a client-server software architecture. Each sensor is a client that connects to the forecasting application (central server), uploading data as needed.

12. One strategy to improve performance is to do all of the authentication and processing in the automated banking machines, and only sending confirmed transactions to the bank server. This would free the bank server from having to deal with authentication, processing and confirmation of requests. On the other hand, improving security would require the bank server to validate all of the processes from the automated banking machines, to prevent compromised automated banking machines from requesting erroneous transactions. Improving performance have a negative effect on security and vice-versa.

13. One way to detect faults would be to identify exceptions in the sensors and their data. The sensors send data periodically, if a sensor has not sent data over a certain period of time then it is faulty. Defining expected ranges for the data can also help in identifying faults, if a sensor is sending .

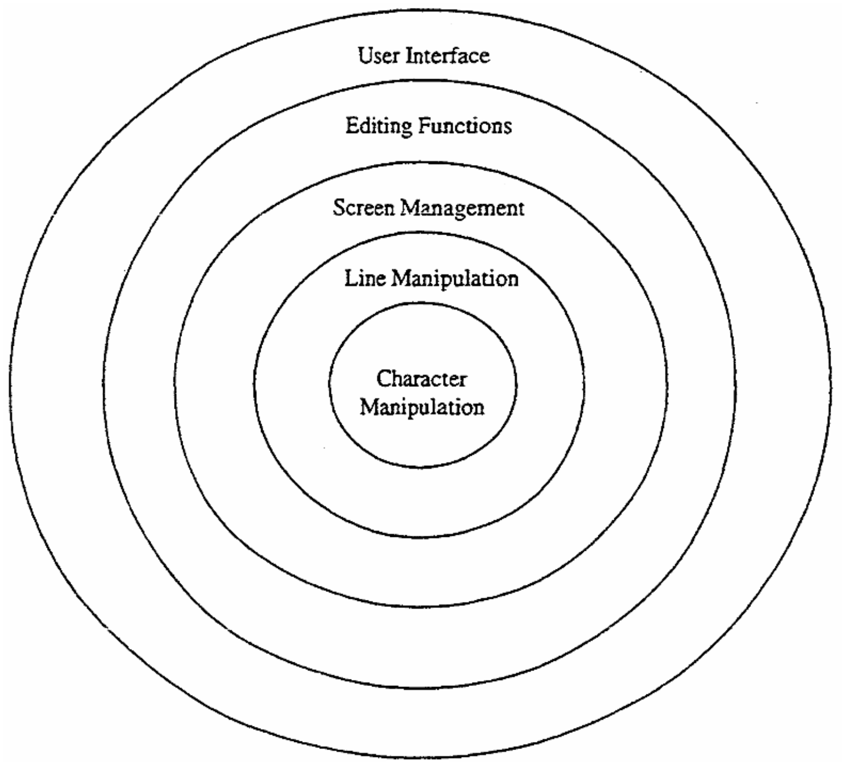
14. The cut set tree is shown below.


Security Breach
Illegal Access Granted
Previous Logout Not Recognized
{Password Exposed, Password Not Changed}

15. It affects the initial cost-benefit analysis by changing the value added by the bin-based indexing. The new value can be calculated by solving the equation
(150 requests/second - 60 requests/second) x 2000/year = $180,000/year.
This modification does not alter the result of the cost-benefit analysis, as design 2 is still the one with the higher ROI and lower payback period.

16. The common graduation requirements for three different disciplines at the (fictional) CGNU are a required minimum average, a minimum number of courses in the area of the discipline, with a subset in specific areas and minimum number of elective courses outside the area of the discipline. The differing requirements are participation in the co-op program for a minimum number of terms for discipline A, enrollment in at least one of the offered lab specializations for discipline B and a final year dissertation for discipline C. These requirements can be combined into a product line, with the common requirements providing the core functionality common to all systems in the family. This product line allows each instantiated application to specialize in a set of discipline specific requirements.

17. The architectural design of the text editor is shown on the next page. The design attempts to reduce coupling by having components in each layer communicate only with each other and with components in adjacent layers. However, there is likely to be a lot of dependency between these components. Other issues to consider when evaluating a more detailed design are cohesion, modularity, complexity, functionality, error and exception handling, and fault handling.


User Interface
Editing Functions
Screen Management
Line Manipulation
Character Manipulation