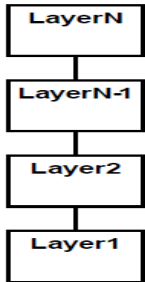
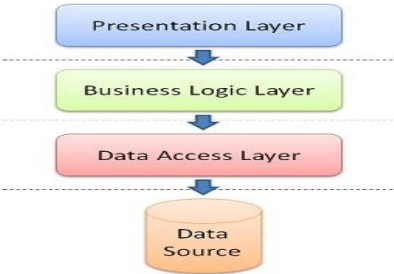
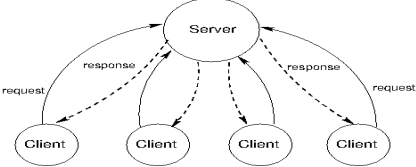
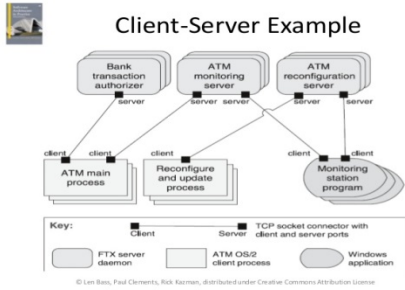
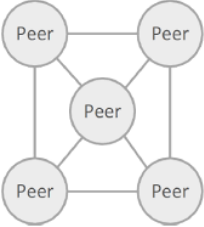
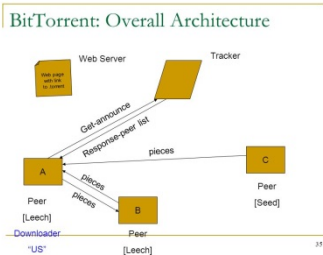
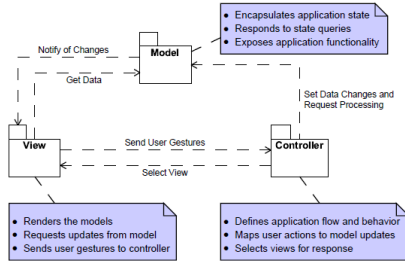
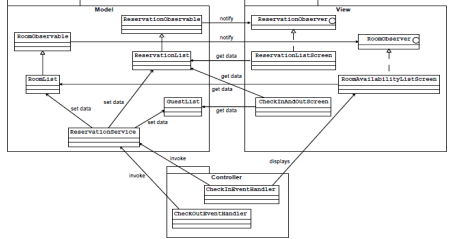
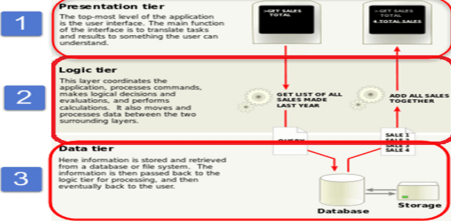
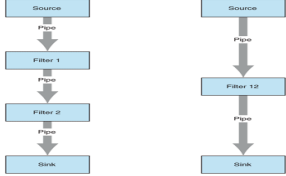

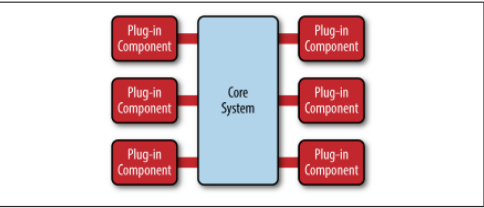
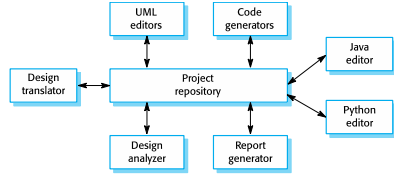


	Pattern	Problem Context	Problem Solution	Example
1	<b>Layered</b>	When we want to partition functionality into different cohesive layers in which a higher layer can request information from a lower layer but a lower layer can only send information (e.g. status) up to a higher layer.		
2	<b>Client Server</b>	When we want to make available a shared set of resources and services e.g. printing or data accessible to a large number of clients who may be distributed across from a range of locations.		 <p>ATM</p>
3	<b>Peer to Peer</b>	When distributed entities need to co-operate and collaborate to provide a service to a community of users e.g. file sharing, or when a computationally intensive problem can be divided into sub-problems and executed more efficiently separately and in parallel.		

4	<b>Model View Controller</b>	When an application involves much user interaction to manipulate and display a lot of data, and we want to make it easy to change the user interface without affecting the underlying data structures and change the underlying data structures without necessarily changing the user interface.	 <ul style="list-style-type: none"> <li><b>Model:</b> <ul style="list-style-type: none"> <li>Encapsulates application state</li> <li>Responds to state queries</li> <li>Exposes application functionality</li> </ul> </li> <li><b>View:</b> <ul style="list-style-type: none"> <li>Renders the models</li> <li>Requests updates from model</li> <li>Sends user gestures to controller</li> </ul> </li> <li><b>Controller:</b> <ul style="list-style-type: none"> <li>Defines application flow and behavior</li> <li>Maps user actions to model updates</li> <li>Selects views for response</li> </ul> </li> </ul>	 <p><b>Hotel Reservation</b></p>
5	<b>N-Tier</b> aka <i>3-Tier, State-Logic-Display</i>	This pattern is another variation on the Layered pattern but “Tiers” are often defined with an eye on the runtime environment, each tier running on an entirely different computer. For example different parts of the system infrastructure may belong to different organisations, or there may be qualitative reasons to do so e.g. performance, security		 <ol style="list-style-type: none"> <li><b>1 Presentation tier</b> The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.</li> <li><b>2 Logic tier</b> This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.</li> <li><b>3 Data tier</b> Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.</li> </ol>
6	<b>Pipe and Filter</b> Data Flow	The Pipe and Filter is pattern is useful for a data processing application in which inputs are processed in separate stages to generate related outputs.		
7	<b>Repository</b> aka <i>Shared Data, Data-Centred, Blackboard, Rule-Based, Microkernel</i>	When an application has large volumes of data that have to be stored for a long time; or is a data-driven systems where the inclusion of data in the repository triggers an action or tool, or where sub-systems must exchange data and there is a large amount of data are to be shared.		 <p><b>Integrated Development Environment</b></p>

<p><b>8</b></p>	<p><b>Event-Based Sense-Compute-Control</b></p>	<p>The Event-Based pattern is useful for event-based applications in which a computer-based system responds to a stream of different events either human or, in embedded control applications from physical sensors which are sampled.</p>		
<p><b>9</b></p>	<p><b>Publish-Subscribe</b></p>	<p>The Publish-Subscribe pattern is useful when many people (Subscribers) want access to similar sets of information that comes from a wide variety of different sources (Publishers). There is a clear delineation between publishers and subscribers.</p>		
<p><b>10</b></p>	<p><b>Service-Oriented Architecture</b></p>	<p>This pattern is useful when needing the continuous delivery of large, complex applications offering a set of relatively independent software services. In Client-Server there is a direct point-to-point connection between the client and the server. In SOA &amp; MSA the connection is via an intermediary component.</p>		

11

**Microservices**

The *Microservices* architecture (MSA) pattern takes the divide and conquer approach of SOA to lower levels of granularity. Service components in MSA are generally single purpose small services that do one thing well whereas in SOA they have more complexity and are often implemented as complete subsystems.

