Layers describe the logical groupings of the functionality and components in an application, whereas tiers describe the physical distribution of the functionality and components on separate servers, computers, networks.

Layers and traditional 3-tier model use the same set of names: presentation, business, services and data, but only tiers imply a physical separation.

Layers are a way of organizing our code, while tiers are only about where the code runs. When we say that our code is hosted on a server it means that our code is hosted on a tier.

The layers of application may reside on the same tier or may be distributed over separate tiers (n-tier) and the components in each layer communicate with component in other layers through well-defined interfaces.

Each tier is completely independent from all other tiers, except from those immediately above and below it. Communication between tiers is typically asynchronous in order to support pattern scalability.

N-tier architectures usually have at least three separate logical parts, each located on a separate physical server. Each part is responsible for specific functionality. When using a layered design approach, a layer is deployed on a tire if more than one service or application is dependent on the functionality exposed by the layer.

Example of tiers: client, application server, database server;

Example of layers: presentation, controller, models, repository.

The easiest solution would be to implement the logging at the top level: Business. And all the layers below should make sure that all necessary failure information gets transported up to the top level.

The implementation of security is made is data layer.