Modularity, layering, and hierarchy represent some of the means to cope with the complexity of a

system. Modularity is a technique to build a complex system from a set of components built and tested

independently. A strong requirement for modularity is to define very clearly the interfaces between

modules and enable the modules to work together.

Modularity has been used extensively since the industrial revolution for building every imaginable

product, from weaving looms to steam engines, watches to automobiles, and electronic devices to

airplanes. Individual modules are often made of subassemblies; for example, the power train of a car

includes the engine assembly, the gear box, and the transmission.

Modularity can reduce costs for manufacturers and consumers. The same module may be used by a

manufacturer in multiple products; to repair a defective product a consumer only needs to replace the

module causing the malfunction rather than the entire product. Modularity encourages specialization

because individual modules can be developed by experts with deep understanding of a particular field.

It also supports innovation, allowing a module to be replaced with a better one without affecting the

rest of the system.

Layering and hierarchy have been present in social systems since ancient times. For example, the

Spartan Constitution, called Politeia, described a Dorian society based on a rigidly layered social

system and a strong military. Nowadays, in a modern society, we are surrounded by organizations that

are structured hierarchically.We have to recognize that layering and hierarchical organization have their

own problems, could negatively affect society, impose a rigid structure and affect social interactions,

increase the overhead of activities, and prevent the system from acting promptly when such actions are

necessary.

Layering demands modularity because each layer fulfills a well-defined function, but the communication

patterns in the case of layering are more restrictive. A layer is expected to communicate only

with adjacent layers. This restriction, the limitation of communication patterns, clearly reduces the

complexity of a system and makes it easier to understand its behavior.

It is no surprise that modularity, layering, and hierarchy are critical to computer and communication

systems. Since the early days of computing, large programs have been split into modules, each with a

well-defined functionality. Modules with related functionalities have then been grouped together into

numerical, graphical, statistical, and many other types of libraries.

Layering helps us deal with complicated problems when we have to separate concerns that prevent

us from making optimal design decisions. To do so, we define layers that address each concern and

design the clear interfaces between the layers.

Probably the best example is layering of communication protocols. In this case it was recognized

that we have to accommodate a variety of physical communication channels that carry electromagnetic,

optical, and acoustic signals; thus, we need a physical layer. The next concern is how to transport bits,

not signals, between two points joined to one another by a communication channel; thus, the need for

a data link layer. Next we decided that we need networks with multiple intermediate nodes when bits

have to traverse a chain of intermediate nodes from a source to the destination; the concern here is how

to forward the bits from one intermediate node to the next, so we had to introduce the network layer.

Then we recognized that the source and the recipient of information are in fact outside the network and

they are not interested in how the information crosses the network, but we want the information to reach

its destination unaltered; the transport layer was then deemed necessary. Finally, the information sent

and received has a meaning only in the context of an application; thus, we needed an application layer.

Layering could prevent some optimizations; for example, cross-layer communication (communication

between non-adjacent layers) could allow wireless applications to take advantage of information

available at the Media Access Control (MAC) sublayer of the data link layer. This example shows that

layering gives us insight as to where to place the basic mechanisms for error control, flow control, and

congestion control on the protocol stack.

An interesting question is whether a layered cloud architecture that has practical implications for

the future development of computing clouds could be implemented. One could argue that it may be

too early for such an endeavor, that we need time to fully understand how to better organize a cloud

infrastructure, and we need to gather data to support the advantages of one approach over another.

On the other hand, there are other systems in which it is difficult to envision a layered organization

because of the complexity of the interaction between the individual modules. Consider, for example,

an operating system that has a set of well-defined functional components:

• The processor management subsystem, responsible for processor virtualization, scheduling, interrupt

handling, and execution of privileged operations and system calls.

• The virtual memory management subsystem, responsible for translating virtual addresses to physical

addresses.

• The multilevel memory management subsystem, responsible for transferring storage blocks between

different memory levels, most commonly between primary and secondary storage.

• The I/O subsystem, responsible for transferring data between the primary memory and the I/O

devices.

• The networking subsystem, responsible for network communication.

The processor management interacts with all other subsystems. There are also multiple interactions

between the other subsystems; therefore, it seems unlikely that a layered organization is feasible in this

case.