A distributed shared memory is an abstraction that hides the details of communicating by sending and receiving messages through a network. The processes cooperate to a common goal by using shared objects (also called concurrent objects). The most famous of these objects is the read/write register, which gives the illusion that the processes access a classical shared memory. Other concurrent objects are the usual objects such as queues, stacks, files, etc.

This part of the book is devoted to the implementation of a shared memory on top of a message-passing system. To that end it investigates two consistency conditions which can be associated with shared objects, namely, atomicity (also called linearizability), and sequential consistency. For a given object, or a set of objects, a consistency condition states which of its executions are the correct ones. As an example, for a read/write shared register, it states which are the values that must be returned by the invocations of the read operation.

This part of the book is made up of two chapters. After having presented the general problem of building a shared memory on top of a message-passing system, Chap. 16 addresses the atomicity (linearizability) consistency condition. It defines it, presents its main composability property, and describes distributed algorithms that implement it. Then, Chap. 17 considers the sequential consistency condition, explains its fundamental difference with respect to atomicity, and presents several implementations of it.