Developing concurrent algorithms requires safety and liveness to be defined in order to understand their proper behavior. Safety refers to the correctness criteria while liveness is the progress guarantee. Nowadays there is a variety of correctness conditions for concurrent objects. The way these correctness conditions differ and the various trade-offs they present with respect to performance, usability, and progress guarantees is poorly understood. This presents a daunting task for the developers and users of such concurrent algorithms who are trying to better understand the correctness of their code and the various trade-offs associated with their design choices and use. The purpose of this study is to explore the set of known correctness conditions for concurrent objects, find their correlations and categorize them, and provide insights regarding their implications with respect to performance and usability. In this survey, a comparative study of Linearizability, Sequential Consistency, Quiescent Consistency and Quasi Linearizability will be presented using data structures like FIFO Queues, Stacks, and Priority Queues, and with a case study for performance of these implementations using different correctness criteria.