Increasing the speed of single-core processors has been facing practical challenges. Instead, multi-core architecture has been ascending for the past decade as the dominant architecture. To gain full advantage of multi-core processors, it is unavoidable for programmers to write concurrent programs. However, writing and reasoning about concurrent programs is often difficult for programmers. One reason for the difficulty stems from the hurdle of dealing with concurrency abstractions, the other reason is the difficulty in getting rid of concurrency related bugs. To address these problems, a new abstraction for concurrent programming has been proposed called capsule. Capsules are inspired by the long-standing ideas explored in the context of actors and other similar models. Although the jury is still out on actors as the de facto abstraction for concurrency, their wide availability and advantages combine to warrant research on the use of this model for concurrency. Capsules explore a new point in this design space to balance flexibility and analyzability in the actor programming model. Unlike common avatars of the actor model, a capsule is statically deployed and configured, confines its local states, permits only a single activity within itself, and communicates with other capsules in a logically synchronous manner. This thesis focuses on the realization, applicability and performance of this new abstraction. A major contribution of this work is the realization of capsules. We have implemented capsules as an extension of javac, the industrial strength standard Java compiler. The implementation shows that it is feasible to extend an object-oriented language with capsules. The default implementation of capsules is based on threads. In this work, we also show alternative implementation strategies to improve the flexibility and adaptability of capsules. This shows that the capsule abstraction can be decoupled from concrete strategies for processing capsule messages and different underlying message processing mechanisms can be deployed without changing the user facing source code. This work also shows the strategy used to retain meta-information about capsules after compilation, so that capsule-oriented programs enjoy the property of separate compilation.