**Exercise 2.76.**

As a large system with generic operations evolves, new types of data objects or new operations may be needed. For each of the three strategies—generic operations with explicit dispatch, data-directed style, and message-passing-style—describe the changes that must be made to a system in order to add new types or new operations. Which organization would be most appropriate for a system in which new types must often be added? Which would be most appropriate for a system in which new operations must often be added?

**Answer.**

We have noticed that the strategy of dispatching on type in section 2.4.2 has two significant weaknesses. One of which is that the interface procedures (`real-part`, `imag-part`, `magnitude`, and `angle`) must know about all the different representations, which forces the interfacde procedures make adaption whenever a new type is added. Another weakness of the technique is that even though the individual representations can be designed separately, we must guarantee that no two procedures in the entire system share the same name. This strategy of non-additive is therefore neither proper for a system in which new types must often be added nor competent for a system in which new operations must often be added.

The data-directed programming style, which handles generic operations in programs by dealing explicitly with operation-and-type tables, is very convenient for adding both new types and new operations into the system without changing any of the existing procedures. This makes it a good choice for both kinds of the frequently changing systems.

In the message-passing-style, a data object is regarded as an intelligent procedure that receives the requested operation and dispatch on the operation name. This indicates that none of existing procedures is required to make any changes when a new type is added into the system, whereas adding a new operation will cause all the former existing procedures to make the corresponding modification. So we can easily conclude that the message-passing-style is useful for a system in which new types must often be added, but will become troublesome if been adopted by a system in which new operations must often be added.