program is contained in only one module. The "depends on" relation does not necessarily define a hierarchy, as two modules may depend on each other either directly or through a longer loop in the "depends on" relation. Note that "depends on" should not be confused with "uses" as defined in a later section.

Information Hiding Structures are the foundation of the object-oriented design paradigm. If an Information Hiding Module is implemented as a class, the public methods of the class belong to the interface for the module.

CONCERNS SATISFIED: The Information Hiding Structures should be designed so that they satisfy changeability, modularity, and buildability.

The Uses Structures

COMPONENTS AND RELATION: As defined previously, Information Hiding Modules contain one or more programs (as defined in the previous section). Two programs are included in the same module if and only if they share a secret. The components of the Uses Structure are programs that may be independently invoked. Note that programs may be invoked by each other or by the hardware (for example, by an interrupt routine), and the invocation may come from a program in a different namespace, such as an operating system routine or a remote procedure. Furthermore, the time at which an invocation may occur could be any time from compile time through runtime.

We will consider forming a Uses Structure only among programs that operate at the same binding time. It is probably easiest first just to think about programs that operate at runtime. Later, we may also think about the uses relation among programs that operate at compile time or load time.

We say that program A uses program B if B must be present and satisfy its specification for A to satisfy its specification. In other words, B must be present and operate correctly for A to operate correctly. The Uses Relation is sometimes known as "requires the presence of a correct version of." For a further explanation and example, see Chapter 14 of Hoffman and Weiss (2000).

The Uses Structure determines what working subsets can be built and tested. A desirable property in the Uses Relation for a software system is that it defines a hierarchy, meaning that there are no loops in it. When there is a loop in the Uses Relation, all programs in the loop must be present and working in the system for any of them to work. Since it may not be possible to construct a completely loop-free Uses Relation, an architect may treat all of the programs in a Uses loop as a single program for the purpose of creating subsets. A subset must include either the whole program or none of it.

When there are no loops in the Uses Relation, a levels structure is imposed on the software. At the bottom level, level 0, are all programs that use no other programs. Level n consists of all programs that use programs in level n–1 or below. The levels are often depicted as a series