2

Frameworks: A Collection of Business Objects

A New Vision

Off-the-shelf computer systems are not always an ideal alternative to in-house built-in systems. It is indeed difficult to find a package that fits all the needs of a health institution. Moreover, the hidden costs involved in customizing a package to integrate it into an existing environment is often outrageous or unknown. There seems to be a trend toward customizing less rather than too much. Major vendors are providing “model” systems to which many early adopters and heavy customers are reverting, often causing unintended effects and maintenance difficulties. On the other hand, custom-made products are disappearing and are gradually being replaced by the assembly of prebuilt reusable components. As software development is similar to the production process of highly sophisticated industrial products, the ultimate goal should be to deliver maintainable and reusable building blocks. Architectures based on components reduce development costs, allow the replacement of components, and finally enable differentiation through customizing. Whenever possible, developers must (re)use existing software components to solve problems that are not related to their domain. In the medical field they should try to reduce the complexity of the medical applications development process and concentrate on the business aspects that ultimately represent the added value of their applications [IBM 1998].

The increasing complexity of medical applications and the need to maintain a high degree of extensibility and at the same time the need to deliver reliable and robust software can be addressed by applying a number of key concepts such as abstraction, decomposition, layering, and iteration. The technique of decomposition, dividing a complex problem into smaller, manageable units, each of which can then be accomplished separately, is the core of component-based development (CBD) [Herzum 2000]. This evolution of application development has also been driven by advances in computer technology and the penetration of Internet and intranet technologies that has changed the way users think about access to information.
Components cover a wide range varying from technological to general and domain-specific aspects [Cummins 2002]. If we analyze commercial admission-discharge-transfer (ADT) systems today, a very large number of functions are the same. The local organization of healthcare services differentiates these packages. Hence, software developers must concentrate on understanding the specific (industry) subdomains together with the corresponding local and cultural requirements.

Before going into more detail, it is important to emphasize that the implementation of any information system follows a logical process from concept definition through requirements and design before arriving at implementation (Figure 2.1).

![Figure 2.1: The software development life cycle.](image)

**Software Components**

There is much discussion about components versus objects and the relationship that exists between them. Software objects are a way to represent the universe of discourse. A software object can be a tangible thing, an idea, or an event. Objects are unique, so they are defined by an identifier. They provide a number of services and allow the manipulation of data in the object. They are made visible to the outside world by means of an interface. The implementation or a description of how the object works is hidden. The implementation consists of data and code (Figure 2.2). Many objects can share the same implementation [Brown 2000]. An object can have many interfaces or many objects can share the same interface.