Abstract: Workflow management systems have attracted a great deal of attention due to their ability to integrate heterogeneous, distributed applications into coherent business processing environments. In spite of their limitations, existing products are enjoying a considerable success but it would be a mistake not to try to see beyond current systems and applications. In today's computer environments, the trend towards using many small computers instead of a few big ones has revived the old dream of distributed computing. There is, however, a significant lack of tools for implementing, operating and maintaining such systems. In particular, there are no good programming paradigms for parallel architectures in which the basic building blocks are stand alone systems. Workflow management provides this key functionality, suggesting its potential as crucial component of any distributed environment. This chapter describes in detail such functionality and provides some insight on how it can be applied in environments other than business processing.

2.1 INTRODUCTION

One of the basic platforms in which to implement generic distributed systems is commodity hardware and software, usually in the form of clusters of workstations connected via a network. The continuous increase in computing power, storage capacity, and communication speed has made these share nothing configurations viable and cost effective alternatives to more tightly integrated multiprocessor architectures. There is also the added advantage of having most of the necessary infrastructure already in place, both in terms of hardware (clus-
ters of personal computers connected by a Local Area Network) and software (the many existing applications). The only component missing in such environments is the necessary glue to make a coherent whole out of many autonomous, heterogeneous, loosely coupled building blocks. This problem has been addressed from many different perspectives, federated database systems [Schaad et al., 1995], TP-monitors [Gray and Reuter, 1993, Obermack, 1994], persistent queuing [Alonso et al., 1995a, Mohan and Dievendorff, 1994], CORBA [OMG, 1995b], process centered software engineering [Ben-Shaul and Kaiser, 1995] and workflow management systems [Hsu, 1995] being among the best examples.

From a practical point of view, these different approaches can be roughly divided in four categories: interface definition, communication, execution guarantees, and development environment. These four categories also correspond to the functionality needed in a distributed environment. In spite of this, existing products and research efforts tend to emphasize only one of the categories, e.g, TP-monitors for execution guarantees; CORBA as an interface definition; queuing systems as communication platforms; or workflow systems for developing distributed applications. Such narrow focus is one of the major limitations of these approaches. Users or designers interested in getting two or more of the four categories of functionality have to resort to combine several heavy-weight solutions, which adversely affects performance and usability. Examples to prove this point abound, perhaps the most clear one being the transactional services described in the CORBA standard. These services can only be implemented using what today is known as a TP-monitor. In fact, current implementations do exactly just that: bundle together a CORBA implementation and a commercial TP-monitor. Since both were designed as stand-alone systems and, in practice, must solve many similar problems, the resulting system incorporates a great deal of redundancy and mismatches. As a result, performance and the overall functionality are adversely affected. A more reasonable approach would be to implement the CORBA standard with the transactional services included as part of the original design instead of as an orthogonal module. This would still not be enough, however, as the resulting system would lack, for instance, a development environment. To address this latter point, the OMG (Object Management Group) and the Workflow Management Coalition are joining efforts to define a CORBA Workflow Facility. But as with the transactional services, such facility will only be truly operational and useful when the design incorporates and integrates all these different technologies from the very beginning and not as separate tools.

This same example occurs in many other environments and products. The underlying problem is that no system incorporates the four categories of functionality in the design and, hence, it is not possible to rely on a truly integrated system. But building such system is only possible if the existing partial solu-