This article presents five run-time architectures for implementing a Workflow Management System (WFMS). The architectures range from highly centralized to fully distributed. Two of the architectures have been implemented at the Large Scale Distributed Information Systems (LSDIS) Lab at The University of Georgia. All the WFMS architectures are designed on top of a Common Object Request Broker Architecture (CORBA) implementation. The article also discusses the advantages and disadvantages of the architectures and the suitability of CORBA as a communication infrastructure. A minor extension to CORBA’s Interface Definition Language (IDL) is proposed to provide an alternative means of specifying workflows. Simplified examples from the healthcare domain are given to illustrate our workflow technology.

Competition and economic pressures force modern business corporations to look for new information technologies to support their business process management. Since workflow technology provides a model for business processes, and “a foundation on which to build solutions supporting the execution and management of business processes” (Hsu and Kleissner, 1995), it has been receiving much attention in the past few years. A workflow can be simply defined as a set of tasks (also called activities or steps) that cooperate to implement a business process. A good workflow technology can also provide a way to make good use of past investments by allowing integration of legacy systems, and the flexibility to support significant organizational changes and technology evolution associated with today’s dynamic enterprises.

A workflow model can be used to design automated or semi-automated solutions for certain business processes within an enterprise and across multiple enterprises. Workflow models tend to be more computer-oriented than traditional business process models. Consequently, they better facilitate automatic generation of substantial portions of actual solutions (i.e., executable workflows).

The history of workflow technology dates back to office automation and batch processing in the late 1970’s, with the first use of the term in early 1980’s (Smith, 1993). In recent years, workflow technology has gained in popularity due to the trend of business process reengineering and many emerging related technologies such as middleware and object-oriented technology, which make the development of a realistic workflow management system possible. A technical overview on workflow management appears in Georgakopoulos et al. (1995) and tutorial materials available include Reinwald (1994) and Sheth (1995). After several years of development, many workflow products and prototypes are now available (White and Fischer, 1994; Georgakopoulos et al., 1995), and early empirical studies in applying the workflow technology or using the products are also being reported (S. Joosten et al., 1994; Juopperi et al., 1995).

A workflow is composed of multiple tasks (also called steps or activities). There are two types of tasks—simple tasks which represent individual indivisible activities, and compound tasks which represent some activities which can be divided into sub-activities (simple tasks or even other compound tasks). An entire workflow can be regarded as a large compound task. A simple task may be a program which can
run on processing entities, which include application systems, servers supported by client-server systems or Transaction Processing Monitors (TP-Monitors), Database Management Systems (DBMSs), etc. Tasks are operations or a sequence of operations that are submitted for execution at the processing entities using their interfaces.

A WorkFlow Management System (WFMS) provides “the ability to specify, execute, report on, and dynamically control workflows involving multiple humans and HAD (Heterogeneous, Autonomous, and Distributed) systems” (Krishnakumar and Sheth, 1995). For workflow execution, a workflow scheduler is necessary to enforce inter-task dependencies, and therefore, to coordinate the execution of tasks in the workflow. Also, task managers are designed to start tasks and to perform a supervisory role in forward recovery.

Most workflow related efforts done in the past few years can be categorized into workflow specification and design, inter-task dependency and scheduling studies, and workflow management system design. Many papers have been published on workflow modeling, workflow specification, and workflow design (Hsu and Kleissner, 1995; Sheth and Rusinkiewicz, 1993; Krishnakumar and Sheth, 1995). Krishnakumar and Sheth (1995) described the modeling and specification of workflows. Forst et al. (1995) proposed a language called C&CO which is an extension of C to specify workflows. Joosten et al performed an empirical study with emphasis on workflow design methods in twelve different organizations (Joosten et al., 1994). Using the term transactional workflows, use of transaction concepts in workflow management was introduced in Sheth and Rusinkiewicz (1993) and subsequently discussed in several papers, including Breitbart et al. (1993), Georgakopoulos and Hornick (1994), Rusinkiewicz and Sheth (1995), Georgakopoulos et al. (1995), Mohan et al. (1995), and Tang and Veijalainen (1995). However, more work needs to be done in this area.

The specification and enforcement of intertask dependencies started with the early efforts to try to specify transaction structure and behavior (Klein, 1991; Chrysanthis and Ramamritham, 1990; Dayal et al., 1991). Klein proposed two primitives to describe conditional existence dependency (Klein, 1991). The task state transition diagram introduced is still a popular way to describe task structures. A variety of database operation related dependencies (e.g., commit dependency and abort dependency) have also been defined (Chrysanthis and Ramamritham, 1992; Ansari et al., 1992; Georgakopoulos and Hornick, 1994).

Attie et al. (1993) formalized inter-task dependencies by using the Computation Tree Logic (CTL) language, introduced a way to synthesize an automaton that captures the computations satisfying the given dependency, and designed a centralized scheduler. Tang and Veijalainen (1995) extended the work in Attie et al. (1993) and proposed a protocol to enforce inter-task dependencies. Several scheduling approaches have been reviewed in Rusinkiewicz and Sheth (1995).

Since workflow management system design depends on workflow specification, research on this aspect started a little later than that of workflow specification. Breitbart et al. (1993) proposed an integrated architecture to support the properties of transactions and their workflow models. The centralized architecture of FlowMark is discussed in Leymann and Altenhuber (1994). Alonso et al. extended the centralized FlowMark architecture to a distributed architecture of a workflow management system (Alonso et al., 1995a). ObjectFlow’s architecture is discussed in Hsu and Kleissner (1995). Kappel et al. presented a WFMS architecture which integrates rules into an object-oriented model (Kappel et al., 1995). Barbara, Mehrotra, and Rusinkiewicz proposed an implementation of a new WFMS model based on the INformation CArrier (INCA) (Barbara et al., 1994). INCAs carry data as well as scheduling/routing information and travel from processing station to processing station. This approach facilitates highly flexible, fully distributed and dynamic workflows, possibly sacrificing efficiency. Issues of low-level infrastructure support for building workflow management systems have been discussed in Georgakopoulos et al., (1995); Reuter and Schwenkreis (1995). So far, we have not seen empirical studies of different workflow system architectures in published literature. An early effort is reported in the thesis by Wang (1995). Workflow reliability and recovery is still a new area where few people have ventured. Discussions on recovery and use of compensation appear in Rusinkiewicz and Sheth (1995); Krishnakumar and Sheth (1995). Failure handling in large scale workflow management systems has been discussed in Alonso et al. (1995b).

Following the advocacy for distributed object management based infrastructure for workflow management systems in Georgakopoulos et al. (1995), this paper discusses the use of specific services provided by PostModern Computing’s ORBeline, a CORBA-based product (we also gained additional experience in using CORBA by using IONA’s Orbix and by participating in a beta test of Sun’s DOE). As a part of our comprehensive investigation into alternative workflow management system architectures, we have designed five runtime architectures. In this paper, we introduce these architectures and specifically comment on the use of ORB services to support these architectures. Task structures play an important role in our workflow model to support heterogeneous nontransactional and transactional tasks. With the aim of easily developing run-time executable code from high level workflow specifications, we propose the Workflow Interface Definition Language (WIDL) that involves a simple extension to CORBA’s IDL.

In section 2 of this article, we discuss the CORBA communication infrastructure and its suitability for workflow. Section 3 briefly describes how workflows are modeled/specified. In section 4, we present the five WFMS run-time architectures. Task structures and task models are presented.
Related Content


Improving Business Intelligence Traceability and Accountability: An Integrated Framework of BI Product and Metacontent Map
[www.igi-global.com/article/improving-business-intelligence-traceability-and-accountability/118087?camid=4v1a](www.igi-global.com/article/improving-business-intelligence-traceability-and-accountability/118087?camid=4v1a)

Incomplete Information in Multidimensional Databases
[www.igi-global.com/chapter/incomplete-information-multidimensional-databases/26972?camid=4v1a](www.igi-global.com/chapter/incomplete-information-multidimensional-databases/26972?camid=4v1a)

Discovering and Analysing Ontological Models From Big RDF Data
[www.igi-global.com/article/discovering-and-analysing-ontological-models-from-big-rdf-data/142072?camid=4v1a](www.igi-global.com/article/discovering-and-analysing-ontological-models-from-big-rdf-data/142072?camid=4v1a)