Event-based Distributed Workflow Execution with EVE

Andreas Geppert, Dimitrios Tombros
Department of Computer Science, University of Zurich
Email: {geppert || tombros}@ifi.unizh.ch

Abstract
In event-driven workflow execution, events and event-condition-action rules are the fundamental metaphors for defining and enforcing workflow logic. Processing entities enact workflows by reacting to and generating new events. The foundation on events facilitates the integration of processing entities into coherent systems. In this paper, we present an event engine, called EVE, implementing event-driven execution of distributed workflows. Its functionality includes event registration, detection and management, as well as event notification to distributed, autonomous, reactive software components which represent workflow processing entities. We describe the distributed, multi-server, multi-client architecture of EVE and its use for workflow execution.

Keywords
event-based systems, workflow management, distributed workflow execution

1 OVERVIEW

Workflow management (Georgakopoulos et al. 1995) has recently found great attention in the information systems field, as it allows to capture knowledge about business processes and to define and enact workflows. **Workflow management systems** (WFMS) are software systems providing workflow definition and execution functionality. **Workflow types** consist of subworkflows and atomic work steps, data flows between them, execution order constraints, as well as assignment of tasks to **processing entities** (PE). The productive use of WFMS requires that they effectively support the integration of heterogeneous information resources, people and applications. The resulting integrated system is called a **workflow system** (WS).

Current commercial WFMS do not effectively support the representation, control, and coordination of PE that operate in distributed environments, that are heterogeneous with respect to their automation degree and programmatic interfaces, that evolve over time, and that operate both independently and as part of a WS. Several research efforts address some of these problems (see section 2), especially those pertaining to distribution and heterogeneity. The proposed solutions, however, do not effectively support flexible representation/integration of PE and runtime evolution of the WS architecture; they also propose a strongly coupled integration framework and rigid task structure.

We propose a **layered event-based** approach providing for a domain-specific architecture model in which PE are represented as autonomous reactive components communicating through parameterized events. The behavior of these components is expressed by **event-condition-action rules** (ECA rules) as provided by active database management systems (ACT-NET 1996). The lowest layer is formed by an event-based middleware layer and execution platform–called EVE–able to integrate these components and make the WS schema executable. Although event-based systems are recognized as the architectural style of choice for loosely coupled systems (Barrett et al. 1996), to the best of our knowledge, no other approach so far uses events as the **only** integration and coordination mechanism for distributed, heterogeneous, and dynamically configurable WS. Event-based WS-architectures and workflow execution have various advantages:

- The event-based coordination model allows complete process specification without imposing limitations on the concrete process architecture. Complex process situations are expressed by composite events, and coordination is accomplished by defining the appropriate reactive behavior for PE.
- The architecture of the WS can be changed during workflow execution (subject to process restrictions).
- A powerful and uniform mechanism to express component behavior is provided; its use for failure and exception handling is straightforward.
- The correct execution and the monitoring of the workflows is guaranteed based on formal semantics (Tombros et al. 1997*b*).

The remainder of this paper is organized as follows. In section 2 we position EVE and survey related work. The WS-architecture model is briefly presented in section 3. We describe the architecture of EVE in section 4 and the realization of distributed event detection and workflow execution in section 5.