Taxonomy of Distributed Event-Based Programming Systems

René Meier and Vinny Cahill
Department of Computer Science, Trinity College Dublin, Ireland
{rene.meier, vinny.cahill}@cs.tcd.ie

Abstract

This paper presents a survey of existing event systems structured as a taxonomy of distributed event-based programming systems. Our taxonomy identifies a set of fundamental properties of event-based programming systems and categorizes them according to the event model and event service criteria. The event service is further classified according to its organization and interaction model, as well as other functional and non-functional features. [1]

1. Introduction

Event-based middleware is currently being applied for application component integration in many application domains including finance, telecommunications, smart environments, multimedia, avionics, health care, and entertainment and event services are omnipresent in applications ranging from small-scale, centralized to large-scale, highly distributed systems. As event-based middleware is exploited in a number of applications in a range of domains, a variety of event services have been proposed to address different application requirements. This paper presents a survey of existing event systems structured as a *taxonomy of distributed event-based programming systems*. Our taxonomy identifies a set of fundamental properties of distributed event-based programming systems, or simply event systems, and categorizes them according to the *event model* and *event service* criteria. The latter is further classified according to its *organization*, *interaction model*, and its *functional* and *non-functional features*. These properties are then arranged in a hierarchical manner starting from the root dimension of the taxonomy, which defines the relationship between event system, event service and event model. Every event system, which we define as an application that uses an event service to carry out event-based communication, has both an event service and an event model. We define an event service as the middleware that implements an event model, hence providing event-based communication to an event system. An event model consists of a set of rules describing a communication model that is based on events. The following sections introduce the event model and the event service dimension of our taxonomy. Figures are presented to outline the relationship among the fundamental properties of event systems and to define the terminology to identify them. Existing event systems are applied to the taxonomy to further outline the identified properties. However, we omit an in dept discussion of our taxonomy and of the given examples due to space limits. A more detailed description and discussion can be found in [5].

[1]The work described in this paper was partly supported by the Irish Higher Education Authority's Programme for Research in Third Level Institutions cycle 0 (1998-2001) and by the FET programme of the Commission of the EU under research contract IST-2000-26031 (CORTEX).

2. Event Model Dimension

The event model defines the application view of an event service. It defines the manner in which an event service is made visible to the application programmer and specifies the components of an event service to which the application programmer is explicitly exposed. Specifically, it classifies the means by which the consuming entities of an application subscribe to the events in which they are interested and the means by which an application raises and delivers events.

We have identified three distinct categories of event model, which are peer to peer, mediator, and implicit. A peer to peer event model allows consuming entities to subscribe at specific named producing entities directly and producing entities to deliver events to specific named subscribed entities directly. For example, the Java distributed event model is based on a peer to peer event model. Event models utilizing a mediator allow consuming entities to subscribe at a designated mediator and producing entities to deliver events to the mediator, which then forwards them to the subscribed consumers. The CORBA event model uses a mediator, called event channel, through which events are propagated. An implicit event model lets consuming entities subscribe to particular event types rather than at another entity or a mediator. Producing entities generate events of some type, which are then delivered to the subscribed con-

sumers. The Cambridge event architecture [1] is based on an implicit event model.

3. Event Service Dimension

The event service dimension deals with the classification of the properties of an event service, which we divide into three distinct categories. The organization sub tree focuses on the distribution of the entities and the middleware of an event system and on the fashion in which the components that comprise an event service cooperate. The interaction model defines the communication path over which producing and consuming entities communicate with each other. It defines the number of intermediate middleware components involved and the manner in which intermediates cooperate to route events from producers to consumers. The feature sub hierarchy addresses the other functional and non-functional features proposed by an event service.

The organization sub tree classifies an event service as either centralized or distributed according to the location of the event system's entities. The entities are centralized if they only reside in the same address space on the same physical machine. In contrast, if the entities of an event system are distributed they may be located in different address spaces possibly on different physical machines. Figure 1 outlines that these two sub categories are further divided exploring the location of the event service middleware. A distributed organization with collocated middleware has been adopted by mSECO [4], which is exclusively located in the same address spaces as the entities. SIENA [2] proposes a set of middleware topologies of which all but the centralized topology use middleware that is distributed over a set of cooperating machines, thus utilizing a distributed organization with separated middleware.

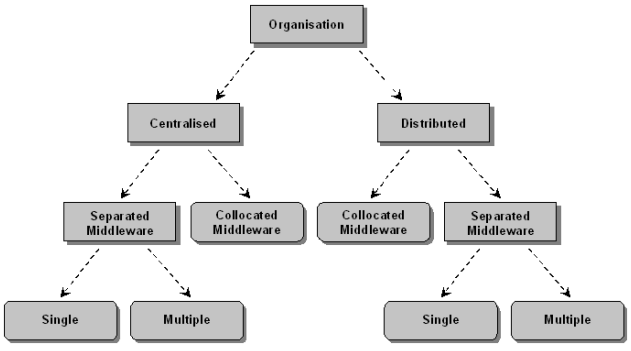

Figure 1. Event Service Organization.

The interaction sub tree classifies an event service according to the interaction model used by the event system. Compared to the organization model, which focuses on the distribution of the entities and the middleware of an event system describing the static view of an event service, the interaction model describes the information flow in a event system. Hence, it describes the dynamic aspect of an event service. As Figure 2 depicts, we divide the interaction model into two main categories, namely intermediate and no intermediate, exploring whether and how many intermediate middleware components an event passes through. Both categories can be divided further. For example, JEDI [3] proposes a hierarchical structure of cooperative distributed intermediates, called dispatching servers, which are interconnected in a tree topology through which events are routed. In contrast, uSECO [4] does not utilize intermediates but uses an name service to resolve the addresses of the entities to which events are routed.

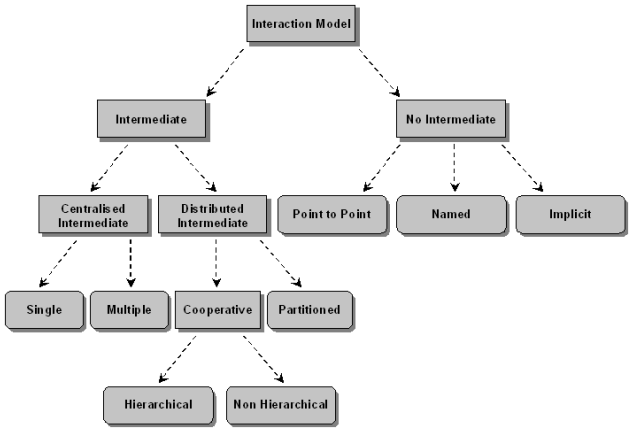

Figure 2. Event Service Interaction Model.

The feature sub hierarchy includes functional features such as event propagation model, event type and filter, mobility and composite events, as well as non-functional features such as QoS, ordering, and fault tolerance.

References

[1] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri. Generic support for distributed applications. *IEEE Computer*, 33(3):68–76, 2000.

[2] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):283–331, 2001.

[3] G. Cugola, E. D. Nitto, and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE TSE*, 27(9):827–850, 2001.

[4] M. Haahr, R. Meier, P. Nixon, V. Cahill, and E. Jul. Filtering and scalability in the ECO distributed event model. In *Proc. of the Int. Symp. on Software Eng. for Parallel and Dist. Systems (PDSE/ICSE2000)*, pages 83–95. Limerick, Ireland, 2000.

[5] R. Meier and V. Cahill. Taxonomy of distributed event-based programming systems. Technical Report TCD-CS-2002, Dept. of Computer Science, Trinity College Dublin, Ireland, March 2002.