Chapter 2
Event Models in Distributed Event Based Systems

Rolando Blanco
University of Waterloo, Canada

Paulo Alencar
University of Waterloo, Canada

ABSTRACT

The event model of a system determines how events are defined and generated, and how events are notified to interested components. In this chapter we look at key differences between the event model in distributed event based systems (DEBSs) and event models found in other implicit invocation systems. We identify features common to all DEBS event models, and variations within different DEBSs implementations. The main goal of the chapter is to elicit important features in event models that need to be supported in the engineering of DEBS applications.

INTRODUCTION

Applications are regularly developed by composing functionality encapsulated in units of computation. Depending on the type of system these units of computation, components for short, include modules, classes, and programs. The functionality provided by components can be composed by procedural abstraction and implicit invocation (Garlan & Shaw, 1994; Dingel et al., 1998; Notkin et al., 1993). When composing functionality by procedural abstraction, also referred to as explicit invocation, names that identify a component are statically bound to the component implementing the functionality. This is the case of a function in one module invoking another function in another module, or a program in one computer using a Remote Procedure Call (RPC) to invoke functionality implemented by a different program on another computer. In contrast, when composing functionality by implicit invocation, a component announces an event. This event announcement triggers the invocation of functionality implemented by another component. The component announcing the event may or may not be required to know the name nor location of the component triggered by the event.

DOI: 10.4018/978-1-60566-697-6.ch002
Distributed Event Based Systems (DEBSs) are implicit invocation systems (IISs) comprised of distributed components that interact with each other using events only. Events are generated by components called publishers. Components interested in the events that have been generated, are called subscribers. A subscriber is notified when an event of interest to the component is generated by a publisher.

The term event model is used in the area of IISs research to characterize two different things. For some researchers, the event model of a system determines the support that the system provides for structuring event data (Rozsnyai et al., 2007). Here, we take a more general approach, similar to that of Garlan and Scott (1993), and Meier and Cahill (2005). We consider the event model to determine the application-level view that a developer must have in order to develop an event-based application or component. Hence, the event model determines how events are defined, how they are announced to other components, how components manifest their interest in events, and how events are delivered to interested components.

The event model found in DEBSs is frequently referred to as publish/subscribe (Eugster et al., 2003). The term publish/subscribe is quite generic and applies, not only to DEBSs, but to any IIS where components invoke a subscribe operation to manifest their interest on events, and a publish operation to announce events. Since the event model of a system goes beyond the actual mechanism used to announce and subscribe to events, we refrain from using the term publish/subscribe, and refer to the event model found in DEBSs, generically, as the DEBS event model.

In this chapter we identify and characterize the features of the DEBS event model, compare the DEBS event model to event models found in other IISs, and illustrate the variations that exist within different DEBSs. The identified features elicit the key aspects that need to be supported by software engineering methodologies, structuring and modularization constructs for DEBSs. Hence, the DEBS event model here presented can be used to validate the support provided by existent software engineering methodologies for the developments of applications in DEBSs, or it can be used as the starting requirement for new proposals.

Understanding the DEBS event model is essential in the development of these systems and their applications. To sustain the need for our categorization, consider the design and specification methods typically proposed for IISs and applied to DEBSs (e.g., Wieringa, 2002). These specification methods assume event models that are incompatible or do not support all the features in the DEBS event model. For example, Harel statecharts (Harel, 1987) assume instantaneous event processing: the reaction to an event occurs in zero time, upon notification of the event. This is not the case when dealing with DEBSs, where events take time to reach subscribed components. Another issue is the assumption that only one event may happen at a time. UML statecharts (OMG, 2007) do not have this restriction, providing instead a queue of events. Unfortunately, both Harel and UML statecharts assume broadcasting of events, where events are globally visible to all components in the system. In contrast, in DEBSs, events are only notified to subscribed components. The incompatibilities between the DEBS event model and the statechart event model make the use of statecharts to model behaviour in DEBSs and DEBS applications difficult. This problem is not unique to DEBSs and arises when dealing with any IISs that have an event model incompatible with the statechart event model. The effect of this impedance between event models has been the proposal of multiple, different, and sometimes incompatible, statechart variations when attempting their use to model complex IISs (e.g., Beeck, 1994; Leveson et al., 1994; Dias & Vieira, 2000; Barbier & Belloir, 2003; Ryu et al., 2006).

Other specification methods based on finite state machines for representing system behaviour also suffer from event model incompatibilities.
Related Content

Extended Action Rule Discovery Based on Single Classification Rules and Reducts
[www.igi-global.com/chapter/extended-action-rule-discovery-based/28473?camid=4v1a](www.igi-global.com/chapter/extended-action-rule-discovery-based/28473?camid=4v1a)

Risk Assessment for Cloud-Based IT Systems
[www.igi-global.com/article/risk-assessment-cloud-based-systems/54192?camid=4v1a](www.igi-global.com/article/risk-assessment-cloud-based-systems/54192?camid=4v1a)

Web Services in Distributed Information Systems: Availability, Performance and Composition
[www.igi-global.com/article/web-services-distributed-information-systems/39039?camid=4v1a](www.igi-global.com/article/web-services-distributed-information-systems/39039?camid=4v1a)

Tag-Based Classification for Software-Defined Networking
[www.igi-global.com/article/tag-based-classification-for-software-defined-networking/128357?camid=4v1a](www.igi-global.com/article/tag-based-classification-for-software-defined-networking/128357?camid=4v1a)