Model-driven Integration of Federated Event Services in Real-time Component Middleware

Gan Deng   Aniruddha Gokhale   Balachandran Natarajan

Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37203

Contact Author: gan.deng@vanderbilt.edu

Abstract

Rapid advances in hardware, networking technologies and software technologies, including standards-based optimized component middleware, has enabled the growth of component middleware-based complex, large-scale distributed real-time and embedded (DRE) systems. These DRE systems found in different domains, such as avionics, telecommunications, defense, enterprise and healthcare, often use a publisher/subscriber communication paradigm, such as that provided by an event service. A federation of such event services provides a scalable solution to address the complex distribution challenges of DRE systems. By connecting event channels from different systems together a federated event service enables seamless and application-transparent interchange of event information across distribution boundaries.

Although component middleware supports the creation of applications via composition of reusable and flexible software components, however, to deploy such systems effectively involves numerous challenges in integrating the various distributed components communicating via different event channels. Current state of the art in deploying a federation of event services for these component middleware-based DRE systems involves ad hoc techniques that are tedious and error-prone.

This paper describes a novel scheme we have developed based on a model-based paradigm that resolves the challenges in configuring the federated event service. Our approach centers around the notion of the federated event service Modeling Language (FESML), which is a modeling tool we have developed to resolve the configuration and deployment challenges of federated event service for component middleware-based DRE systems.

Key Words: Federated Event Service, Component Middleware, CORBA Component Model, Model-based systems.

1 Introduction

Over the past decade, *component middleware* has evolved to support the creation of applications via composition of reusable and flexible software *components*. Components are implementation/integration units with crisply-defined interfaces that can be installed and instantiated in application server run-time environments. Middleware is systems software that resides between applications and the underlying operating systems, network protocol stacks, and hardware in complex distributed systems to enable or simplify how these components are connected. Examples of commercial-off-the-shelf (COTS) component middleware include the CORBA Component Model (CCM) [1], J2EE [2], and .NET [3]. Large-scale Distributed Real-time and Embedded (DRE) systems (such as avionics mission computing [4], distributed audio/video processing [5], and distributed interactive simulations [6]), which is the focus of our research, are increasingly based on component middleware.

One model of communication between components supported by component middleware is based on the publish/subscribe paradigm [7]. This is achieved via the integration of event or notification services within the component middleware. The publisher/subscriber design is a powerful architecture for event-based communication because it provides anonymity, by decoupling event publishers and subscribers, and asynchronism, by automatically notifying subscribers when a specified event is generated.

DRE systems are often characterized by the presence of a large number of components using the publish/subscribe communication paradigm. Examples of these systems include distributed interactive simulation environments, such as the Next-Generation Run Time Infrastructure (RTI-NG) implementation for the Defense Modeling and Simulation Organization (DMSO) High Level Architecture (HLA) [8].

Naive implementations of an event service will send one message for each remote consumer interested in the event. This design will waste network resources since the same data is transmitted multiple times, often to the same target host.

The strategy by which a predictably performing distributed event service with minimal network traffic overhead can be configured within our optimized CCM implementation called CIAO [9] is to build a federation of real-time event services. Such a service, called a federated event service, allows sharing filtering information to minimize or eliminate the transmission of unwanted events to a remote entity. Moreover, a federated event service allows events that are being communicated in one channel to be made available on another channel. The channels typically communicate through CORBA Gateways, UDP, or IP Multicast [10]. By connecting event channels from different systems together a federated event service allows event information to be interchanged seamlessly, providing a level of integration between the two systems, thus improving the quality of service (QoS) of the system.

To deploy a federated event service for a complex DRE system requires complex integration efforts in configuring a federation of event services with a distributed component middleware. Current state of the art in deploying such a federation involves *ad hoc* integration efforts based mostly on handcrafting the federation deployment descriptor metadata. This metadata is usually specified in XML adhering to some DTD or Schema. The descriptor metadata comprises information, such as the type of federations (*i.e.*, using a CORBA Gateway, or UDP or IP Multicast), object references of remote event channels, object reference of local event channels, and other information. Since the components may involve a large number of different types of events and event channels, *ad hoc* techniques of deploying federation of event services is a very tedious and error-prone task.

To address this challenge, therefore, requires principled methods that can be analyzed, validated and verified for correctness and robustness. Model-based techniques are well suited to address these challenges. Therefore, this paper provides a novel approach we have developed using model-based techniques to deploy federation of event services for DRE systems.

Previous work on federated event services has focused on the patterns and performance optimizations of *highly scalable* [6] and *real-time* [11] CORBA Event Service [12] implementations in the context of real-time CORBA object middleware [10]. This paper extends this previous work by describing how our ongoing R&D on Model Driven Middleware (MDM) [13] can be applied to to simplify the integration of a federated event service in QoS-enabled component middleware.

MDM is an emerging paradigm that integrates *model-based software techniques* (including Model-Integrated Computing [14, 15] and the OMG's Model Driven Architecture [16]) with *QoS-enabled component middleware* (including Real-time CORBA [8] and QoS-enabled CCM [9]) to help resolve key software development and validation challenges encountered by developers of large-scale DRE middleware and applications.

Section 2 describes a MDM tool we have developed to simplify the configuration and deployment of federated event service in CIAO, which is our QoS-enabled implementation of the CORBA Component Model; and Section 3 provides concluding remarks and future work.

2 Resolving Federated Event Service Integration Challenges in CIAO

To address the challenges in deployment of federated event services described in Section 1, we have developed the Federated Event Service Modeling Language (FESML), as a part of of the CoSMIC [17] tool chain [18], which is a Model Driven Middleware toolsuite that supports the development, assembly, and deployment of DRE systems. The artifacts that FESML provides include event consumers, event suppliers, event channels, CORBA Gateways, UDP Senders, UDP Receivers and Multicast ports, among others.

Figure 1 illustrates an example of how FESML can be used to model a federation of event channels using CORBA gateways and which includes other artifacts of the publish/subscribe paradigm, such as event consumers, event suppliers, event channels in different sites.

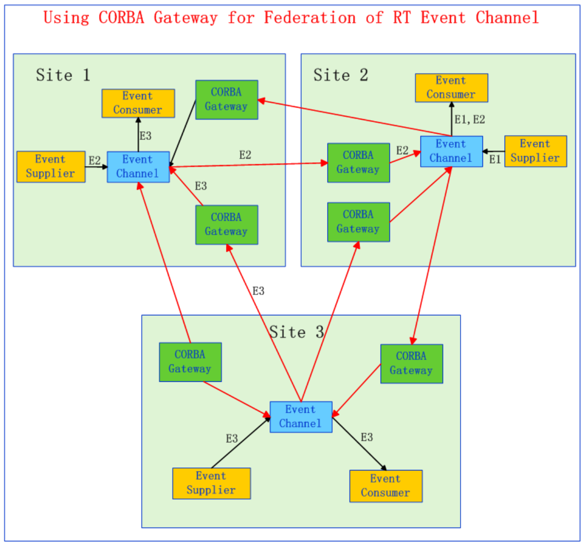

Figure 1: **Federated Event Channel**

The rest of the section describes the artifacts of the FESML modeling paradigm.

2.1 FESML Syntactic Elements

The FESML meta-model defines two level of syntactic elements: (1) The outer level contains only the *Site* element, which allows the user to define how many sites are present in the distributed environment and how they could be connected with each other through event channels and CORBA Gateway, which are exposed as *Port* elements from the outside view; (2) The second level, which is the inner level representing a site, contains a list of syntactic elements including Event Supplier, Event Consumer, Event Channel, CORBA Gateway, IP Multicast Sender and Receiver, and Event Type References, which allows the user to configure the deployment of these artifacts inside a site.

Figure 2 is a screenshot of the FESML tool used to model a federated event service with three sites. The top part of the Figure 2 shows the outer level configuration, and the bottom part shows the inner level of the configuration.

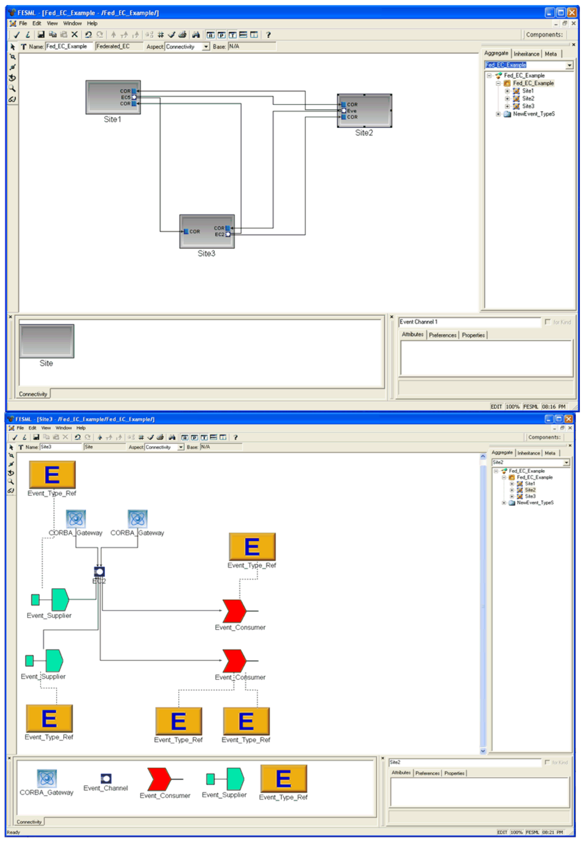

Figure 2: **FESML Artifacts**

2.2 Model Checker

To ensure the validity of the modeled federated event service, the event channels' configuration and settings must be checked to ensure that they are consistent with the federation types. For example, when the user chooses the IP Multicast as the type of federations to federate event channels, since IP Multicast uses the Observer [19] capabilities of event channels, the Observer functionality in event channel must be enabled and activated to ensure the IP Multicast could work properly. To ensure such semantically consistent configuration, it would be ideal that any violation of such rules will be detected in the early modeling phase rather than in the later component deployment phase.

FESML provides a built-in constraint model checker that checks for syntactic and semantic compatibility of the federation of event channels to ensure the correct assembly and deployment of event service.

2.3 Model Interpreter

The FESML encompasses a model interpreter, which synthesizes the federated event service assembly and deployment descriptor XML files. The information captured in the descriptor files includes the relationship between each artifacts, the physical location of each supplier, consumer, event channel, CORBA Gateway, etc. This file will be then further fed into the CIAO assembly and deployment tool to deploy the system.

Currently there is no standard XML DTD or Schema to describe a real-time event service configuration metadata or metadata for the federation of the event services. Our approach, therefore, extends the existing standard component assembly metadata DTD with elements for deployment of federated event services.

The shift toward high-level design languages and modeling tools is creating an opportunity for increased automation in generating and integrating application components. The goal is to shield all the low level details about how to configure the federation of event service away from application developers.

3 Concluding Remarks and Future Work

Large-scael distributed real-time and embedded systems need a real-time publisher/subscriber paradigm for communication. This entails the need to deploy a federation of event services. This paper describes a novel approach of using model-based techniques to deploy a federated event service. We have developed a tool called FESML to address the assembly and deployment challenges of federated event services.

The true test of FESML will be its usefulness in configuring real component-based software systems. In the coming months, this tool will be tested in multiple real-world scenarios involving mission-critical distributed, real-time systems.

The OMG has issued an specification for a Notification Service, which is a superset of the CORBA Event Service that adds interfaces for event filtering, configurable event delivery semantics, security, and event delivery at specified levels of QoS. CIAO already provides the implementation for the Notification Service specification. FESML is being enhanced to allow these services to be modeled and configured in a large distributed environment.

References

[1] Object Management Group, *CORBA Components*, OMG Document formal/2002-06-65 edition, June 2002.

[2] Sun Microsystems, "Java^{TM} 2 Platform Enterprise Edition," http://java.sun.com/j2ee/index.html, 2001.

[3] Microsoft Corporation, "Microsoft .NET Development," msdn.microsoft.com/net/, 2002.

[4] Christopher D. Gill, David L. Levine, and Douglas C. Schmidt, "The Design and Performance of a Real-Time CORBA Scheduling Service," *Real-Time Systems, The International Journal of Time-Critical Computing Systems, special issue on Real-Time Middleware*, vol. 20, no. 2, Mar. 2001.

[5] David A. Karr, Craig Rodrigues, Yamuna Krishnamurthy, Irfan Pyarali, and Douglas C. Schmidt, "Application of the QuO Quality-of-Service Framework to a Distributed Video Application," in *Proceedings of the 3rd International Symposium on Distributed Objects and Applications*, Rome, Italy, Sept. 2001, OMG.

[6] Carlos O'Ryan, Douglas C. Schmidt, and J. Russell Noseworthy, "Patterns and Performance of a CORBA Event Service for Large-scale Distributed Interactive Simulations," *International Journal of Computer Systems Science and Engineering*, vol. 17, no. 2, Mar. 2002.

[7] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal, *Pattern-Oriented Software Architecture—A System of Patterns*, Wiley & Sons, New York, 1996.

[8] Arvind S. Krishna, Douglas C. Schmidt, Ray Klefstad, and Angelo Corsaro, "Real-time CORBA Middleware," in *Middleware for Communications*, Qusay Mahmoud, Ed. Wiley and Sons, New York, 2003.

[9] Nanbor Wang, Douglas C. Schmidt, Aniruddha Gokhale, Craig Rodrigues, Balachandran Natarajan, Joseph P. Loyall, Richard E. Schantz, and Christopher D. Gill, "QoS-enabled Middleware," in *Middleware for Communications*, Qusay Mahmoud, Ed. Wiley and Sons, New York, 2003.

[10] Douglas C. Schmidt and et al., “TAO: A Pattern-Oriented Object Request Broker for Distributed Real-time and Embedded Systems,” *IEEE Distributed Systems Online*, vol. 3, no. 2, Feb. 2002.

[11] Douglas C. Schmidt and Carlos O'Ryan, “Patterns and Performance of Real-time Publisher/Subscriber Architectures,” *Journal of Systems and Software, Special Issue on Software Architecture - Engineering Quality Attributes*, 2002.

[12] Object Management Group, *Event Service Specification Version 1.1*, OMG Document formal/01-03-01 edition, Mar. 2001.

[13] Aniruddha Gokhale, Douglas C. Schmidt, Balachandran Natarajan, Jeff Gray, and Nanbor Wang, “Model Driven Middleware,” in *Middleware for Communications*, Qusay Mahmoud, Ed. Wiley and Sons, New York, 2003.

[14] Janos Sztipanovits and Gabor Karsai, “Model-Integrated Computing,” *IEEE Computer*, vol. 30, no. 4, pp. 110–112, Apr. 1997.

[15] Jeffery Gray, Ted Bapty, and Sandeep Neema, “Handling Crosscutting Constraints in Domain-Specific Modeling,” *Communications of the ACM*, pp. 87–93, Oct. 2001.

[16] Object Management Group, *Model Driven Architecture (MDA)*, OMG Document ormsc/2001-07-01 edition, July 2001.

[17] Center for Distributed Object Computing, “Component Synthesis using Model Integrated Computing (CoSMIC),” www.dre.vanderbilt.edu/cosmic, Vanderbilt University.

[18] Aniruddha Gokhale, Krishnakumar Balasubramanian, Jaiganesh Balasubramanian, Arvind Krishna, George T. Edwards, Gan Deng, Emre Turkay, Jeffrey Parsons, and Douglas C. Schmidt, “Model Driven Middleware: A New Paradigm for Deploying and Provisioning Distributed Real-time and Embedded Applications,” *Submitted to The Journal of Science of Computer Programming: Special Issue on Model Driven Architecture*, 2004.

[19] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA, 1995.