Model-based Performance Prediction for Event-driven Systems*

Christoph Rathfelder
FZI Forschungszentrum Informatik
76131 Karlsruhe
Germany
rathfelder@fzi.de

Samuel Kounev
FZI Forschungszentrum Informatik
76131 Karlsruhe
Germany
kounev@fzi.de

ABSTRACT

The event-driven communication paradigm provides a number of advantages for building loosely coupled distributed systems. However, the loose coupling of components in such systems makes it hard for developers to estimate their behavior and performance under load. Most existing performance prediction techniques for systems using event-driven communication require specialized knowledge to build the necessary prediction models. In this paper, we propose an extension of the Palladio Component Model (PCM) that provides natural support for modeling event-based communication and supports different performance prediction techniques.

1. INTRODUCTION

In Event-Driven Architectures (EDA), system components communicate by sending and receiving events. Compared to synchronous communication using for example remote procedure calls, this decoupled communication between components promises several benefits including more loosely-coupled services and better scalability. However, the event-driven programming model is more complex, as application logic is distributed among multiple independent event handlers and the flow of control during execution is harder to track. This increases the complexity of modeling event-driven architectures for performance prediction in the early phases of system development.

Performance modeling and prediction techniques, surveyed in [2], support the architect in evaluating different design decisions. However, most existing performance prediction techniques for systems using event-driven communication require specialized knowledge to build the necessary prediction models (e.g., [1]). Furthermore, general purpose design oriented performance models for component-based systems provide limited support for modeling event-driven communication.

*This work was supported by the European Commission (grant No. FP7-216556)

DEBS'09, July 6–9, Nashville, TN, USA.

In this paper, we present an extension of the Palladio Component Model (PCM) [3] that eases the design-oriented modeling of event-driven component-based systems. We then describe how the newly introduced modeling constructs can be mapped to existing PCM constructs. Using an automated model-to-model transformation, this allows to reuse existing prediction techniques supported by PCM while significantly reducing the modeling effort and complexity.

2. PALLADIO COMPONENT MODEL

The Palladio Component Model (PCM) [3] is a domain-specific modeling language for modeling component-based software architectures. It supports automatic transformation of design-oriented architectural models to analysis-oriented performance models including layered queuing networks [5], stochastic process algebras and simulation models. In PCM, architectural models are parameterized over the system usage profile and execution environment.

Software components are the core entities of PCM. They contain an abstract behavioral specification called Resource Demanding-Service Effect Specification (RD-SEFF) for each provided service. Similar to UML activities, RD-SEFFs consist of three types of actions: internal actions, external service calls, and control flow nodes including branches, loops and forks. *Internal actions* model resource demands abstracting from computations performed inside components. *External service calls* represent component invocations of services provided by other components. *Branches* represent "exclusive or" splits of the control flow. *Loops* model the repetitive execution of a set of actions. *Forks* split the control flow into multiple concurrently executing threads.

PCM currently supports only synchronous communication between components following the call-return communication style. We now present an extension of PCM that provides natural support for modeling event-driven systems.

3. MODEL EXTENSIONS

In PCM, it is possible to use a combination of non-synchronized fork actions and external service calls to model asynchronous communication. As shown in [4], this workaround allows to model components communicating asynchronously over message queues. However, given that with this approach asynchronous events are modeled using synchronous service calls, a semantic gap between the system implementation and the architecture model is introduced. Moreover, the modeling effort of this approach dramatically increases if event-driven communication following the publish-subscribe paradigm is considered. To reduce this overhead and to

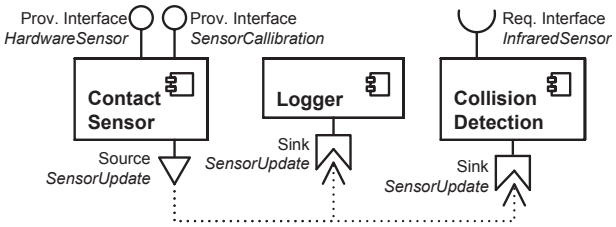

Figure 1: Example Scenario

eliminate the semantic gap, it is necessary to extend PCM with the following elements allowing to model event-driven communication explicitly:

- **Events** are the central element of event-driven communication. In contrast to interfaces which include method signatures, events only specify the underlying data type. For example, a *SensorUpdateEvent* can be defined as complex data type that includes the sensor ID, a timestamp and the new and old value of the sensor reading. This allows to consider the event data when modeling a component's behavior by means of an RD-SEFF.

- **Event Sources** specify that a component emits a certain type of events. For each emitted event type, the component must provide a respective event source. Furthermore, it is necessary to extend the RD-SEFF with a new action called **Event Action** allowing to instantiate and send events.

- **Event Sinks** specify that the component receives and processes certain types of events. In analogy to the event sources, each consumed event type induces a separate event sink. Only compatible event sources and sinks are allowed to be connected. Each event sink requires the specification of an **Event Handler**. Event handlers are modeled similar to ordinary component services using RD-SEFF.

Figure 1 shows a simplified part of an event-driven component-based system. The `ContactSensor` component provides an interface to calibrate the sensor and another one (used by the hardware controller) to set the actual sensor value. Additionally, the component provides an event source for `SensorUpdate` events. The `Logger` component consumes `SensorUpdate` events from different types of sensors including contact sensors and persists them in a database. The `CollisionDetection` component consumes `SensorUpdate` events from the `ContactSensor` and is used to detect collisions with the help of an additional infrared sensor.

In the following, we present a mapping of the introduced model extensions to existing model elements in the current version of PCM to enable the reuse of supported performance prediction methods. For the sake of brevity, we focus on the most important elements and provide the complete transformed model as a download[1]. Event sinks are transformed into interfaces provided by the respective component. Each interface includes the service `OnEvent` with the respective event type as input parameter. Additionally, it is possible to integrate for example the marshalling of events as

[1]http://palladio-approach.net

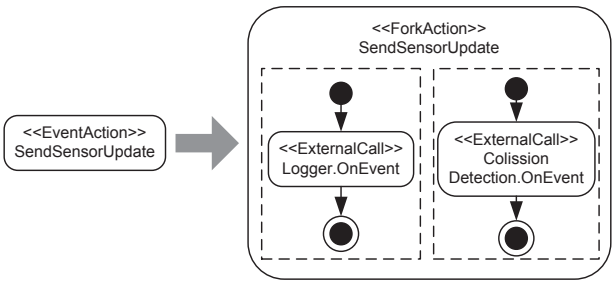

Figure 2: Transformation of an EventAction

presented in [4] for point-to-point connections. The transformation of event sources and the associated event actions is more complex and requires much more modeling effort if it is done manually. Figure 2 illustrates the mapping of an event action into a fork action which includes an external service call for each connected event sink. Furthermore, it is necessary to explicitly require an interface for each connected source, because PCM supports only 1-to-1 connections between required and provided interfaces.

4. ONGOING AND FUTURE WORK

The proposed extensions of PCM allow a semantically correct modeling of event-driven systems. In combination with a model transformation following the proposed mapping, they significantly reduce the effort to build and analyze models by means of the analytical and simulative performance prediction techniques supported in PCM. The automation of this transformation is part of our current work. Furthermore, we plan to introduce some further constructs in PCM to support modeling of an event bus. In our future work, we will study the performance-relevant influence factors associated with event-driven communication. As a first step, we intend to focus on the influence of persistent vs. non-persistent delivery, the number of event consumers, and the event filtering mechanisms. Based on these results, we plan to extend the models and respective transformations to consider these factors with the aim to increase the prediction accuracy.

5. REFERENCES

[1] R. Baldoni, M. Contenti, S. Piergiovanni, and A. Virgillito. Modeling publish/subscribe communication systems: towards a formal approach. pages 304–311, Jan. 2003.

[2] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-Based Performance Prediction in Software Development: A Survey. *IEEE Transactions on Software Engineering*, 30(5):295–310, May 2004.

[3] S. Becker, H. Koziolek, and R. Reussner. The Palladio component model for model-driven performance prediction. *Jour. of Syst. and Softw.*, 82:3–22, 2009.

[4] J. Happe, S. Becker, C. Rathfelder, H. Friedrich, and R. H. Reussner. Parametric Performance Completions for Model-Driven Performance Prediction. *Performance Evaluation*, 2009. Accepted for publication in 2009.

[5] H. Koziolek and R. Reussner. A Model Transformation from the Palladio Component Model to Layered Queueing Networks. In *SIPEW 2008*, 2008.