Towards Middleware-aware Integration of Event-based Communication into the Palladio Component Model*

Christoph Rathfelder
FZI Research Center for Information Technology
Karlsruhe, Germany
rathfelder@fzi.de

Benjamin Klatt
FZI Research Center for Information Technology
Karlsruhe, Germany
klatt@fzi.de

Samuel Kounev
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
kounev@kit.edu

David Evans
University of Cambridge
Cambridge, UK
david.evans@cl.cam.ac.uk

ABSTRACT

The event-based communication paradigm is becoming increasingly ubiquitous as an enabling technology for building loosely-coupled distributed systems. However, the loose coupling of components in such systems makes it hard for developers to predict their performance under load. Most general purpose performance meta-models for component-based systems provide limited support for modelling event-based communication and neglect middleware-specific influence factors. In this poster, we present an extension of our approach to modelling event-based communication in the context of the Palladio Component Model (PCM), allowing to take into account middleware-specific influence factors. The latter are captured in a separate model automatically woven into the PCM instance by means of a model-to-model transformation. As a second contribution, we present a short case study of a real-life road traffic monitoring system showing how event-based communication can be modelled for performance prediction and capacity planning.

1. INTRODUCTION

In component-based systems using event-based communication, system components interact by sending and receiving events. Compared to synchronous communication using, for example, remote procedure calls (RPCs), event-based communication among components promises several benefits including more loosely-coupled services and better scalability. However, the event-based 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. Furthermore, the middleware used for communication influences the performance of the whole system. This increases the difficulty of modelling component-based systems with event-based communication for performance prediction.

In [5, 6], we described an extension of the Palladio Component Model (PCM) [2] allowing to model event-based communication and analyse the system performance by means of a model-to-model transformation. PCM is a design-oriented performance meta-model for modelling component-based software architectures. It allows to explicitly capture component context dependencies (e.g., dependencies on the component usage profile and execution environment) and provides support for a number of different performance analysis techniques. The model transformation we sketched in [5] did not distinguish between platform-independent and platform-specific influence factors. More precisely, the logical flow of events between components was mixed with platform-specific resource demands and middleware-specific behaviours.

In this poster, we present a refined transformation based on our work in [5, 6]. The transformation provides a clear separation between platform-independent and platform-specific influence factors. This is achieved by capturing the behaviour of the communication middleware into a separate model that is automatically woven into the PCM model instance. The proposed approach allows to easily evaluate and compare the performance influences of different middleware implementations, as only the platform-specific repository has to be replaced without changing the transformation. In addition to this transformation, we present the results of a case study showing how our modelling approach can be applied to a road traffic monitoring system based on the event-based middleware SBUS (Stream BUS) [3].

2. MODEL TRANSFORMATION

The performance analysis and simulation techniques integrated in the PCM tool chain currently do not support event-based communication. To enable a semantically correct modelling of systems using event-based communica-

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

DEBS'10, July 12–15, 2010, Cambridge, UK

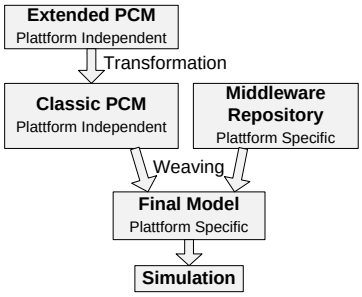

Figure 1: Model Transformation Process

tion, we extended the PCM with new modelling constructs (e.g., EventSource, EventSink, EventAction) [6]. As already shown in [5], a transformation of these extensions into existing PCM modelling constructs is required in order to be able to use the existing model analysis techniques.

To realize the above mentioned separation of platform-independent and platform-specific influence factors, we divided the transformation into two parts. As shown in Fig. 1, the extended PCM model, that includes the new constructs enabling a semantically correct modelling of event-based communication, is first transformed into a generic model representing the event-based communication using a combination of ExternalCallActions and Forks as sketched in [6]. This model does not include platform-specific details like resource demands or sizes of thread-pools. In the next step, a platform-specific component model, located in a separated repository, is woven into the PCM model instance. The platform-specific models have to be created only once and can be reused in different contexts. The result of the transformation is a model instance which includes platform-specific details and can be analysed by means of analytical or simulation techniques. The transformation is performed automatically and is fully transparent. It is only required to select the component repository that corresponds to the used middleware.

Fig. 2 shows the result of the platform-independent part of the transformation. For each Sink and Source an additional component is added. These composite components include several components that represent different steps in the middleware's event processing:

- **SourcePort** Interaction between source component and middleware.
- **DistributionPreparation** Processing within the middleware, that is done once per event (e.g. marshalling before the distribution).
- **EventDistribution** Splitting the control flow and distribution the events to all sinks.
- **EventSender** Processing within the middleware that is done for each connected sink (e.g. communication handshake).
- **EventReceiver** Processing of the event within the middleware, but located on the receiver side. (e.g. Receiving the event from the communication channel or demarshalling)
- **SinkPort** Communication between middleware and receiving component.

Each of these internal components includes an ExternalCall to the predefined interfaces `IMiddlewareSource` or `IMiddlewareSink`. These interfaces and ExternalCalls are later connected with the platform-specific repository and the respective components included in the repository.

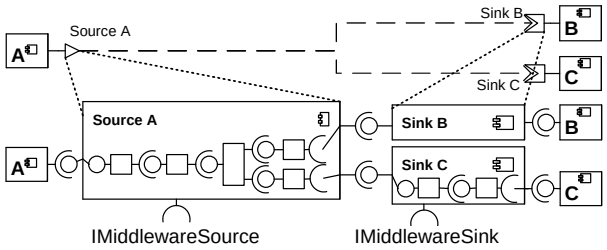

Figure 2: Platform-independent Result of the Transformation

3. CASE STUDY

To evaluate the applicability of our modelling approach as well as the accuracy of the prediction results, we applied our modelling approach to a case study of a real-life road traffic monitoring system [4]. The system we studied is developed as part of the TIME project (Transport Information Monitoring Environment) [1] at the University of Cambridge. The system is based on the SBUS middleware [3] which supports peer-to-peer event-based communication including both continuous streams of data (e.g., from sensors), asynchronous events, and synchronous RPC. We considered a number of different scenarios under different deployment topologies and load conditions. We measured CPU utilization as well as processing times within components. We compared the measurements with the prediction values and for all scenarios, the prediction error was less then 10% in most of the cases and did not exceed 20%.

4. ONGOING AND FUTURE WORK

The proposed transformation allows a separation of generic event-based communication and middleware specific characteristics. The fully automated integration of this transformation into the PCM tool chain is part of our current work. Additionally, we are working on an extended case study using the SBUS middleware and we are planning to conduct a separate case study using a centralized messaging system based on the Java Message Service (JMS). The goal will be to demonstrate how the effect of using different implementations of JMS for event-based communication can be reflected in the PCM model by incorporating different middleware repositories.

5. REFERENCES

[1] J. Bacon, A. R. Beresford, D. Evans, D. Ingram, N. Trigoni, A. Guitton, and A. Skordylis. TIME: An open platform for capturing, processing and delivering transport-related data. In *Proc. of the IEEE consumer communications and networking conference*, 2008.

[2] 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.

[3] D. Ingram. Reconfigurable middleware for high availability sensor systems. In *Proc. of DEBS'09*.

[4] C. Rathfelder, D. Evans, and S. Kounev. Predictive Modelling of Peer-to-Peer Event-driven Communication in Component-based Systems. In *Proc. of ECSA 2010*, 2010. under review.

[5] C. Rathfelder and S. Kounev. Fast Abstract: Model-based Performance Prediction for Event-driven Systems. In *Proc. of DEBS'09*, 2009.

[6] C. Rathfelder and S. Kounev. Modeling Event-Driven Service-Oriented Systems using the Palladio Component Model. In *Proc. of QUASOSS'09*, 2009.