Modeling of Event-based Communication in Component-based Architectures

Christoph Rathfelder, Benjamin Klatt and Samuel Kounev

Keynote talk presented by Samuel Kounev at FESCA @ ETAPS 2012
Tallinn, Estonia, March 31, 2012
Agenda

- Introduction to Event-based Communication
- Design-time Modeling and Analysis
  - Palladio Component Model (PCM)
  - Meta-Model Extensions for Event-based Communication
  - Case Studies
- Run-Time Quality-of-Service Management
  - Descartes Meta-Model (DMM)
- Outlook
References (1)

References (2)


Agenda

- Introduction to Event-based Communication
- Design-time Modeling and Analysis
  - Palladio Component Model (PCM)
  - Meta-Model Extensions for Event-based Communication
  - Case Studies
- Run-Time Quality-of-Service Management
  - Descartes Meta-Model (DMM)
- Outlook
Decoupling in the 3 dimensions [Eugster]

Communicating parties:

1. Do not need to be active at the same time (Time)
2. Do not need to know each other (Space)
3. Are not blocked when exchanging messages (Synchronization)
Event-based Communication (2)

- **Event**
  - A significant change in state

- **Source**
  - Producer, publisher, sender, generator, or monitoring component.

- **Transmission System**
  - Notification service, event service, event-based middleware, channel or event bus.

- **Sink**
  - Reactive components, consumers, subscribers, or receivers
Event Delivery Model

(a) Point-To-Point

(b) Channel-based Publish/Subscribe
Motivating Example
Store Scenario

Source: UpdateStockData

Sink: UpdateStockData

Event Bus

RFID Scanner

Cashdesk Service

Order Management Service

Inventory Management Service

Logging Service

Prov. Interface: CreateOrder

Req. Interface: CreateOrder

© Christoph Rathfelder, Benjamin Klatt and Samuel Kounev
Motivating Example
Store Scenario

Changing Usage

VS.

Prediction

• Max. Throughput
• Processing Time

Analyses

• Bottleneck
• Resource Utilization

Sizing

VS.
Agenda

- Introduction to Event-based Communication
- Design-time Modeling and Analysis
  - Palladio Component Model (PCM)
  - Meta-Model Extensions for Event-based Communication
  - Case Studies
- Run-Time Quality-of-Service Management
  - Descartes Meta-Model (DMM)
- Outlook
Palladio Software Quality Prediction Framework

Palladio Component Model
- Domain specific modelling language
- Aligned with UML 2 design models
- Component-based architectures
- http://www.palladio-approach.net

Eclipse-based modelling and prediction tool
- Design-time quality prediction
- http://www.palladio-simulator.com

Transformations into prediction models
- Simulation code
- Layered queueing networks
- Queueing Petri nets
Palladio Modelling Approach
Model Solution Techniques

- Component Model
- Architecture Model
- Deployment Model
- Usage Model

Palladio Approach

Service-Level Prediction
- Simulation Code
- Layered Queueing Networks
- Queueing Petri Nets
- Stochastic Regular Expr.

Resource Utilization

Response Time

Model Solution Techniques

- Palladio Approach
- Component Model
- Architecture Model
- Deployment Model
- Usage Model

Service-Level Prediction
- Simulation Code
- Layered Queueing Networks
- Queueing Petri Nets
- Stochastic Regular Expr.

Resource Utilization

Response Time

Christoph Rathfelder, Benjamin Klatt and Samuel Kounev

© 31.03.2012

Descartes Research Group
Institute for Program Structures and Data Organization
Agenda

- Introduction to Event-based Communication
- Design-time Modeling and Analysis
  - Palladio Component Model (PCM)
  - Meta-Model Extensions for Event-based Communication
- Case Studies
- Run-Time Quality-of-Service Management
  - Descartes Meta-Model (DMM)
- Outlook
Integration of Event-Based Communication in the Palladio Software Quality Prediction Framework

Benjamin Klatt, FZI
Christoph Rathfelder, FZI
Samuel Kounev, KIT

Events and Components
Direct Point-to-Point Connections
Graphical Notation

Interface

Component

Component

Component

Component

Component

Event Source

Event Sink

Component

Component

Component

Component
Publish/Subscribe Communication
PCM Meta-Model Extensions

- Explicit modelling of
  - Events
  - Source and sink ports
  - Many-to-many connections
  - Event producer (Source) and handler (Sink)

Sources, Sinks & Connectors

Components & Roles

Emit Event Action

© Christoph Rathfelder, Benjamin Klatt and Samuel Kounev
Approach

Quality-Prediction of component-based architectures with event-based communication integrated into Palladio

- Explicit event modeling with reduced effort
- Considering event communication middleware influences
- Reuse existing prediction techniques
1. Meta-Model Extension

Extension of the PCM meta-model

Semantically correct modelling of

- Events
- Source and sink ports
- 1-many connections
- Event handlers
2. Model-to-Model Transformation

Extensions are substituted with existing elements
Performance equivalent model
Allows reuse of existing prediction techniques
Does not consider platform-specific resource demands
3. Weaving of Platform Specific Components

- Middleware repository
  - Platform-specific components
    - Behaviour
    - Resource demands
- Weaving with platform-independent model
- Input for quality predictions
Transformation Example
Agenda

- Introduction to Event-based Communication
- Design-time Modeling and Analysis
  - Palladio Component Model (PCM)
  - Meta-Model Extensions for Event-based Communication
- Case Studies
- Run-Time Quality-of-Service Management
  - Descartes Meta-Model (DMM)
- Outlook
Motivating Example

- Traffic Monitoring in Cambridge

![Diagram of traffic monitoring system including traffic lights, sensors, and storage location.]

© Christoph Rathfelder, Benjamin Klatt and Samuel Kounev
Motivating Example

Traffic Monitoring in Cambridge

Design Alternatives
- One component per
  - Traffic light
  - Intersection
  - District
  - New components
  - Speeding detection

Changing Usage
- TrafficLight

Prediction
- Processing Time
- Resource Utilisation

Deployment and Sizing

Analyses
- Max. Throughput
- Bottleneck
- Event delivery latency

© Christoph Rathfelder, Benjamin Klatt and Samuel Kounev
Preliminary Case Study - Application

Traffic Monitoring in Cambridge

- Output of TIME-EACM research project
- Real world scenario

- SBUS (Stream Bus) middleware
  - Supports RPC as well as event streams
  - Developed in Cambridge

Project website: http://www.cl.cam.ac.uk/research/time/
Preliminary Case Study – System Model

<<System>>

- SBUS Case Study

- Acis
- SBUSSource5
- SBUSSink7
- Location Storage
- ThreadPool5
- ThreadPool8
- Scoot
- SBUSSource2
- SBUSSink10
- Redlight
- ThreadPool4
- ThreadPool11
- SBUSSource3
- SBUSSink12
- SBUSServer1.5
- SBUSClient1.4
Preliminary Case Study – Evaluation

- Prediction of
  - Event processing time
  - CPU utilisation
- Different scenarios
  - Several instances
  - Different deployment
  - Up to 4 quad-core machines
  - Variation of event rates
- Prediction error
  - Mostly < 10%
  - Never exceeded 20%
Preliminary Case Study – Evaluation (2)

Initial Modelling Effort

Model Adaptation Effort
Capacity Planning for Event-based Systems using Automated Performance Predictions

- Christoph Rathfelder
  *FZI, Germany*
- Samuel Kounev
  *KIT, Germany*
- David Evans
  *University of Cambridge*

Motivating Example

Event Bus

- Cam
- Cam
- License Plate Recognition
- Toll
- Bus Sensors
- Traffic Control
- Location
- Bus Proximity
- Speeding

© Christoph Rathfelder, Benjamin Klatt and Samuel Kounev
Motivating Example

Changing Load

System Evolution

- New components
- New algorithms

Sizing and Capacity Planning

- Max. throughput
- Resource utilization
- Latency
- Bottlenecks

Over-provisioning of hardware resources
Capacity Planning using Performance Predictions

System Modeling/Resource Demand Estimation

Capacity Planning

System Deployment/Reconfiguration

System Evolution/Workload Change

End of life?

Yes

Model Adaptation

Performance Predictions

QoS requirements fulfilled?

No

Yes

Resources used efficiently?

Yes

Capacity Planning

Variation of Architecture/Deployment
Automated Performance Prediction Process

1. Systematic variation of simulated workload intensity
2. Automated model transformations
3. Simulation-based performance prediction using PCM
Transformation Example

Platform independent

Source A
- SourcePort
- Distribution Preparation
- Event Distribution
- Event Sender

IMiddleware SourcePort
IMiddleware DistributionPreparation
IMiddleware EventDistribution
IMiddleware EventSender

Platform specific

SBUS SourcePort
SBUS Middleware

IMiddleware Receiver
IMiddleware SinkPort

Sink C
- Event Receiver
- SinkPort

IMiddleware Receiver
IMiddleware SinkPort

Middleware-Weaving

M2M-Event-Transformation

Parameter Variation

End of Parameter Range?

Solving/Simulation

Transformation Prediction Model

© Christoph Rathfelder, Benjamin Klatt and Samuel Kounev
Case Study

- Traffic Monitoring System
  - Based on output of TIME-EACM research project
  - Real world scenario with data from the City of Cambridge
- SBUS (Stream Bus) middleware
  - Supports RPC as well as event streams
  - Developed in Cambridge
- Scenarios
  - System variations
  - Evaluation of deployment options
    - Maximal throughput
    - Hardware utilization
    - Latency
    - Bottlenecks
### Specification of Architecture-level Prediction Model

<table>
<thead>
<tr>
<th>Component Repository</th>
<th>1 Variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Controlled experiments for each component</td>
<td></td>
</tr>
<tr>
<td>• Resource demand estimation based on time measurements</td>
<td></td>
</tr>
<tr>
<td>• Probabilistic and parameter dependent</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System Model</th>
<th>3 Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Instantiation and connection of components</td>
<td></td>
</tr>
<tr>
<td>• Variations depending on scenario (e.g., load balancing)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deployment and Hardware</th>
<th>7 Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Specification of hardware resources</td>
<td></td>
</tr>
<tr>
<td>• Deployment of components depending on scenario</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage Model</th>
<th>&gt; 100 Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Automated variation of workload</td>
<td></td>
</tr>
</tbody>
</table>
Scenario 2 - Growing Workload

- Adding additional cameras causes additional workload
- License Plate Recognition (LPR) is bottleneck

**Result:** Small improvement by deploying LPR on dedicated server
Scenario 3 - Additional Component

- New Toll component
  - LPR is the bottleneck
- 2 additional servers
- Load balancing on 3 LPR instances

**Result:** Centralized deployment of Speeding and Toll
Scenario 4 – Improved Cameras

- New cameras with higher resolution
  - Improved LPR success rate
  - Higher overall CPU demand for processing

**Result:** Decentralized deployment of Speeding and Toll

---

Old cameras

New cameras

© Christoph Rathfelder, Benjamin Klatt and Samuel Kounev
Evaluation of Prediction Accuracy

- Deployment of each scenario in our testbed
- Load drivers with configurable event rate using real world data
- Compare measured and predicted values in different load situations

Each machine equipped with: Intel Core 2 Quad Q6600 2,4GHz, 8GB RAM, Ubuntu 8.04
Evaluation Results

Scenario 3

- LPR Meas. (decent.)
- LPR Pred. (decent.)
- LPR Meas. (cent.)
- LPR Pred. (cent.)
- Proc. Meas. (cent.)
- Proc. Pred. (cent.)

Scenario 4

- Meas. (cent., old)
- Pred (cent., old)
- Meas. (decent., old)
- Pred. (decent., old)
- Meas. (cent., new)
- Pred (cent., new)
- Meas. (decent., new)
- Pred (decent., new)

Prediction error:
- Utilization always underestimated
- Mean error < 20%, Max error < 25%

31.03.2012  © Christoph Rathfelder, Benjamin Klatt and Samuel Kounev
Descartes Research Group
Institute for Program Structures and Data Organization
Effort Reduction

Architecture-level prediction models

- Eased variation of architecture and deployment

Automated model transformation for events

- 80% less manual element creations compared to manual modeling

Automation of performance predictions

- Time saving
  - Prediction time: 3 min
  - Experiment run time: 2.7 hours
- Automated load-variation
Conclusion

Capacity Planning

- Based on automated performance predictions
- Prediction error < 25%
  - Always underestimated
- Improves the system‘s efficiency
  - Often over-provisioning by factor 2 and more

Effort Reduction

- Modeling effort for event-based systems reduced by 80%
- Significant time saving by using prediction techniques
Agenda

- Introduction to Event-based Communication
- Design-time Modeling and Analysis
  - Palladio Component Model (PCM)
  - Meta-Model Extensions for Event-based Communication
- Case Studies
- **Run-Time Quality-of-Service Management**
  - Descartes Meta-Model (DMM)
- Outlook
Descartes Meta-Model (DMM)

- Architecture-level modeling language for self-aware run-time systems management of modern IT systems, infrastructures and services

- Main Goal: Provide Quality-of-Service (QoS) guarantees
  - Performance (current focus)
    - Response time, throughput, scalability and efficiency
  - Or more generally, dependability
    - Including also availability, reliability and security aspects
1) **Self-Reflective**
   Aware of their software architecture, execution environment and hardware infrastructure, as well as of their operational goals

2) **Self-Predictive**
   Able to anticipate and predict the effect of dynamic changes in the environment, as well as the effect of possible adaptation actions

3) **Self-Adaptive**
   Proactively adapting as the environment evolves to ensure that their operational goals are continuously met

http://www.descartes-research.net
Descartes Meta-Model (DMM)

Collection of several meta-models each focusing on different system aspects
PCM and DMM

Palladio Component Model (PCM)  Descartes Meta-Model (DMM)

Design-time aspects  Run-time aspects

© Christoph Rathfelder, Benjamin Klatt and Samuel Kounev
Design-time vs. Run-time Models

- Two orthogonal dimensions
  - Modeling of design-time vs. run-time aspects
  - Use of models at design-time vs. run-time

- Fine granular differentiating factors
  1. Model purpose
  2. Model target users / consumers
  3. Degrees of freedom in model use case scenarios
  4. Model structure & parameterization
  5. Possibilities for model calibration
  6. Required model flexibility
Descartes vs. Palladio

DMM

PCM

© Christoph Rathfelder, Benjamin Klatt and Samuel Kounev
Example Usage Scenarios

Example: Design-time QoS analysis

Example: Run-time QoS management

Example: Elasticity evaluation at design-time

Example: Design-time QoS analysis in a DMM resource landscape
The Descartes Meta-Model

Samuel Kounev, Pakistan Brosig, Nikolaus Huber
Descartes Research Group
Chair for Software Design & Quality (SDQ)
Karlsruhe Institute of Technology (KIT), Germany
kounev@kit.edu brosig@kit.de huber@kit.de
March 21, 2012
v0.50

http://descartes.ipd.kit.edu/research_and_profile/descartes_meta_model/
Discussion

http://www.descartes-research.net/