Event Stream Processing Units in Business Processes

S. Appel, S. Frischbier, T. Freudenreich, A. Buchmann

Databases and Distributed Systems Group
TU Darmstadt

appel@dvs.tu-darmstadt.de
http://www.dvs.tu-darmstadt.de

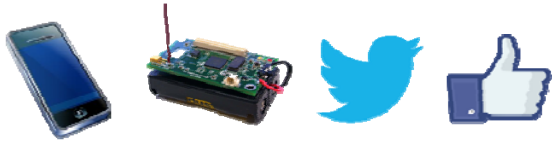
Event Stream Processing Units in Business Processes

S. Appel, S. Frischbier, T. Freudenreich, A. Buchmann

Databases and Distributed Systems Group
TU Darmstadt

appel@dvs.tu-darmstadt.de
http://www.dvs.tu-darmstadt.de

Event Stream Integration enhances Business Processes

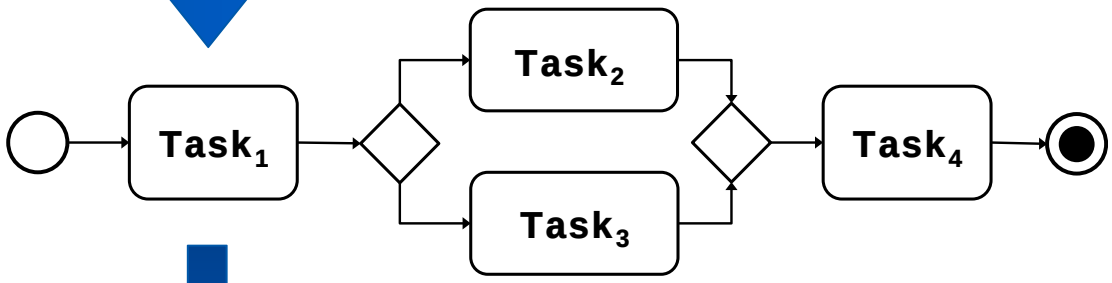


- Event Streams are ubiquitous
- Integration of real world feedback in business processes:

→ Enhanced processes

→ New business cases

- *Example*: Logistics process with shipment monitoring



BPM World and the Event-based World

TECHNISCHE UNIVERSITÄT DARMSTADT

BPM World

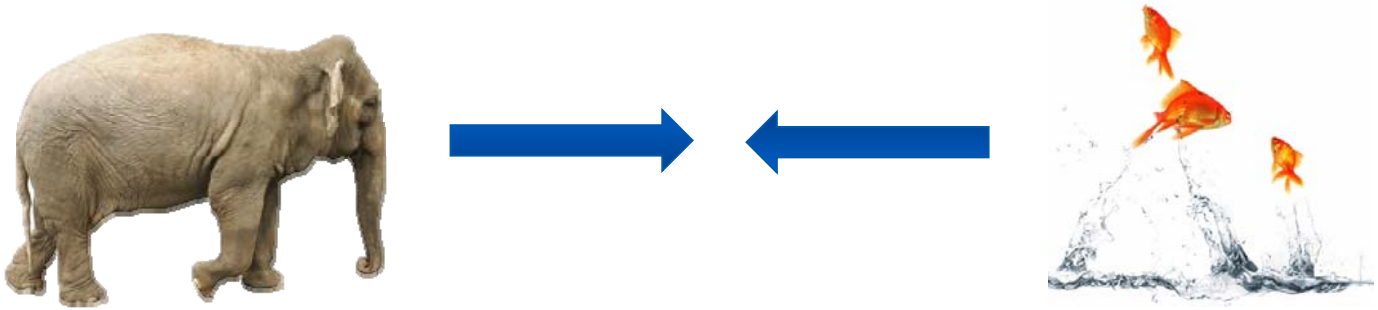
- Structured business processes
- Top-down view
- Pull-based approach with explicit invocation

  SOA, Databases, ERP, etc.
- Data processing: Request/reply

Event-based World

- Real-time data streams consisting of events
- Bottom-up view
- Push-based approach with implicit invocation

  Complex Event Processing etc.
- Data processing: Reactive and asynchronous (pub/sub)



Integration of Event Stream Processing in BPM Workflow

TECHNISCHE UNIVERSITÄT DARMSTADT

BP Modeling (BPMN)

Transition

BP Execution (BPEL)

Interaction

IT Infrastructure (SOA)

DVS

Business Functions: Coherent Abstraction across Layers

TECHNISCHE UNIVERSITÄT DARMSTADT

BP Modeling (BPMN)

BPMN Task

BP Execution (BPEL)

Service Invocation

IT Infrastructure (SOA)

Web Service

Business Function

DVS

Event Stream Processing Encapsulation as Business Function

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Event Stream Processing Unit (SPU):**
  Container for event stream processing at the abstraction level of a business function
- Main SPU characteristics across layers:
  - Implicit invocation possible (triggered by event)
  - Continuous operation
    → no request/reply
    → requires implicit and explicit completion
  - Input data not known at invocation
    → publish/subscribe
- Need to be addressed across the modeling, execution, and IT layer

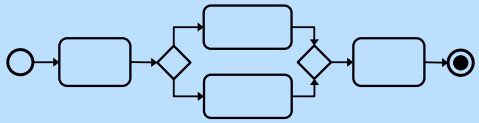
DVS

SPU Realization across Layers

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Modeling Layer

- BPMN notation for SPUs
- Execution semantics for continuous operation
- Data streams as input and output



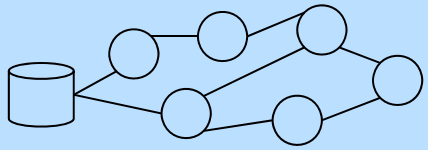
2. Execution layer

- Process execution control flow with support for implicit and explicit invocation
- Data input as subscriptions to events

```
<process name="pns:Caller">
  <invoke partnerLink= "Link">
    <service name="wns:Billing" />
  </invoke>
</process>
```

3. IT Layer

- Service-like container model to encapsulate event stream processing as business functions

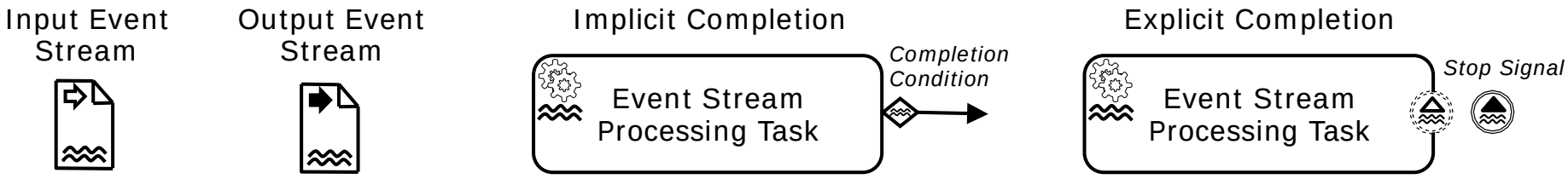


DVS

1. Modeling Layer: BPMN 2.0 Extension

TECHNISCHE UNIVERSITÄT DARMSTADT

- Event Stream Specification (ESS)
  - Input Event Stream → Subscription
  - Output Event Stream → Advertisement
- Event Stream Processing Task (ESPT)
  - Continuous operation
  - Explicit completion: triggered externally via signal
  - Implicit completion: evaluated internally

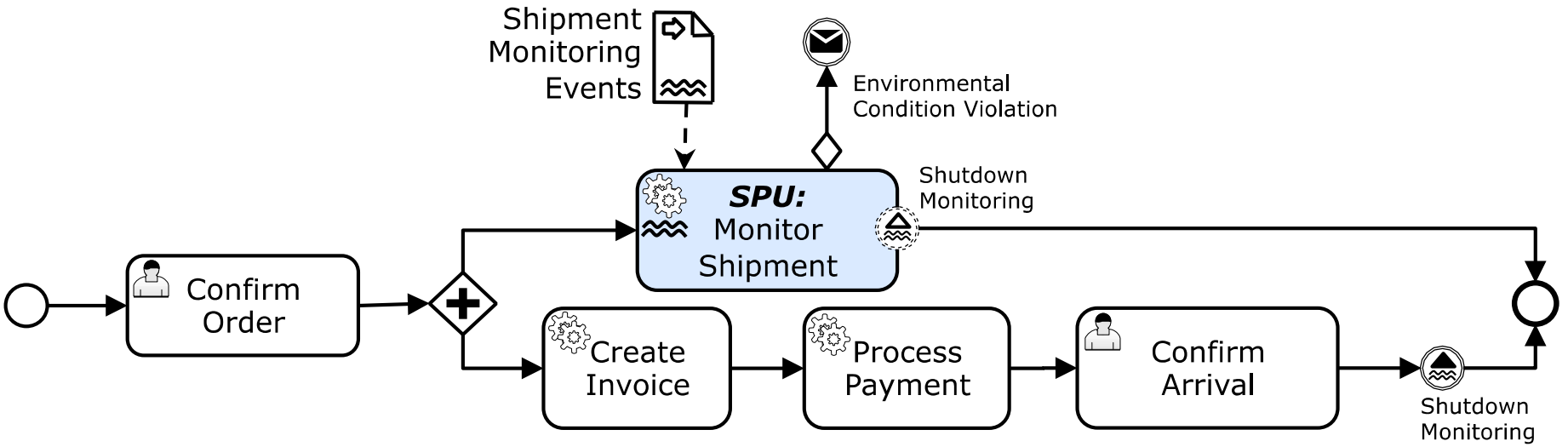


DVS

1. Modeling Layer: Task with Explicit Completion

TECHNISCHE UNIVERSITÄT DARMSTADT

- ESPT completion triggered by signal from within process
  - Responsibility of process execution engine
  - Completion when event processing is known to be not required anymore
  - Completion is controlled, e.g., persisting data, closing connections

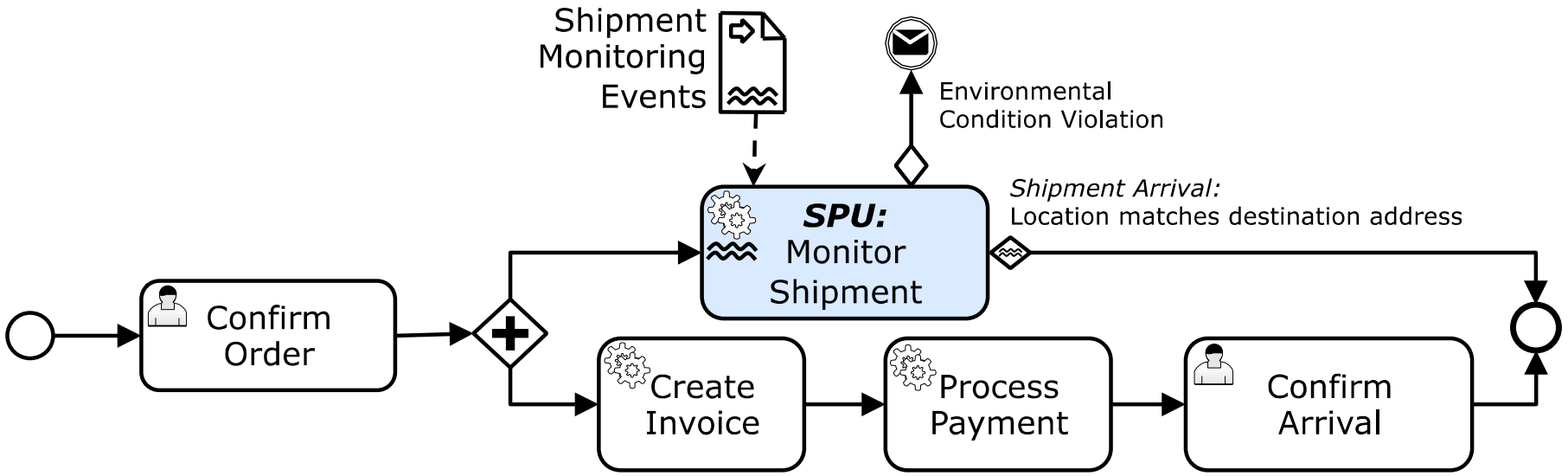


DVS

1. Modeling Layer: Task with Implicit Completion

TECHNISCHE UNIVERSITÄT DARMSTADT

- ESPT completion triggered from task application logic implementation
  - Responsibility of IT infrastructure
  - Completion when condition is met (timeout, event pattern, etc.)
  - Back channel to process execution engine

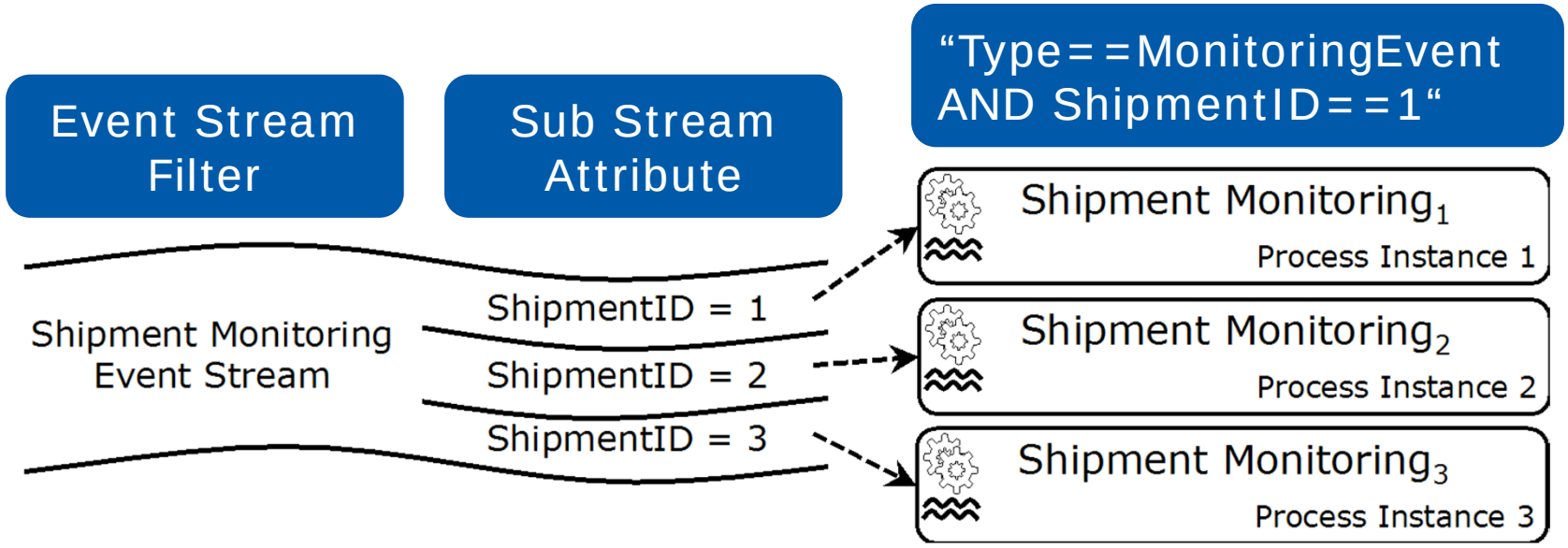


DVS

2. Execution Layer: Runtime View

TECHNISCHE UNIVERSITÄT DARMSTADT

- Event stream processing unit instance per entity instance (e.g., per shipment)
  - At execution layer, specification of SPU input to derive subscription
  - *Event Stream Filter:* General filter, applies to all SPU instances
  - *Sub Stream Attribute:* Identifies entity instance event streams

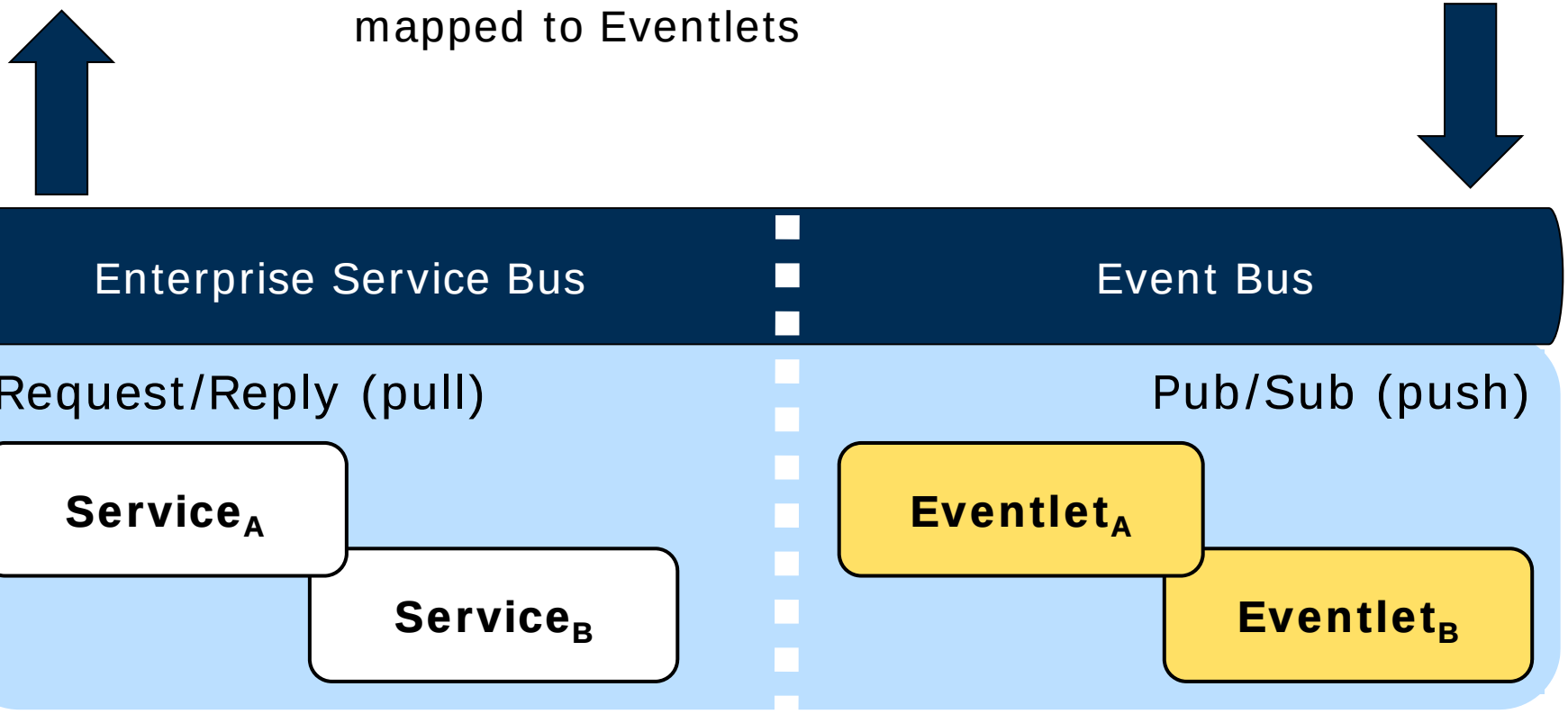


DVS

3. IT Layer: Eventlets as service-like containers for event stream processing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Service Tasks are mapped to web services
- Event Stream Processing Tasks are mapped to Eventlets



3. IT Layer: Shipment Monitoring Example

TECHNISCHE UNIVERSITÄT DARMSTADT

Shipment Monitoring

Eventlet Metadata

CompletionCondition:	*Timeout(120sec)*
EventStreamFilter:	*MonitoringEvent*
SubStreamAttribute:	*ShipmentID*

<Shipment42>

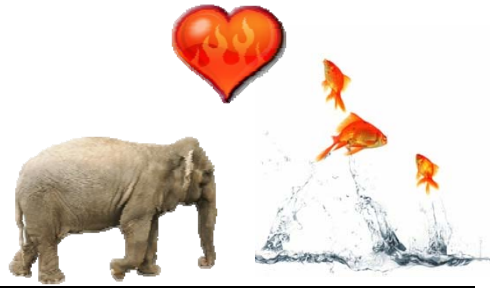
Eventlet Runtime Code

```
onInstantiation(subStreamId id) {
 limit = getTempThreshold(id); }

onEvent(Event e) {
 if (e.getValue("temp") > limit)
     raiseAlert(); }

onRemove() {}
onCompletion() {}
```

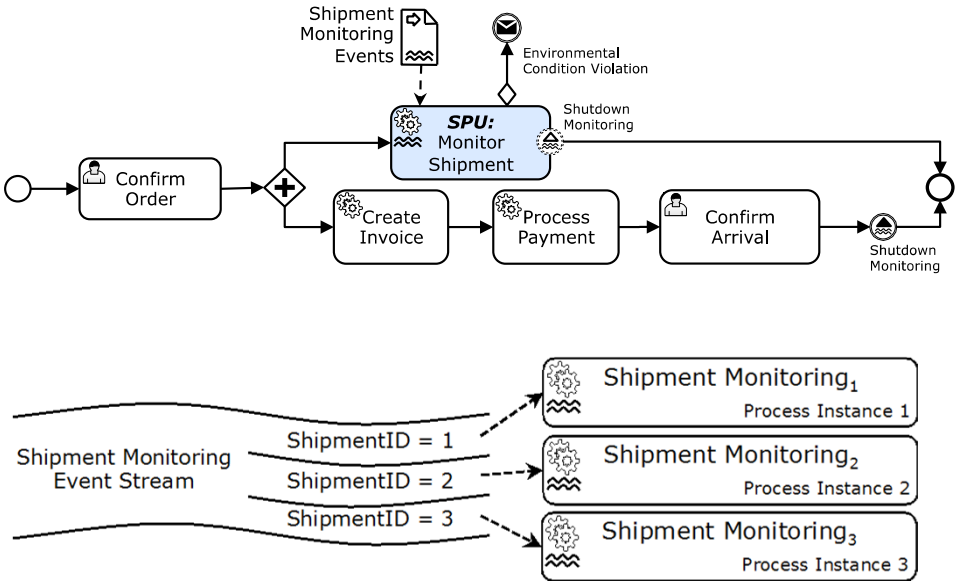
Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Event Stream Processing Units (SPUs)** encapsulate event stream processing as business function
- SPUs ensure a high coherence across modeling, execution, and IT layer
- SPUs at modeling layer: **Event Stream Processing Tasks** as BMPN extension with implicit and explicit completion semantics
- SPUs at execution layer: Support of implicit and explicit instantiation by process execution engine
- SPUs at IT layer: **Eventlets** – Container model for event stream processing



Questions and Comments



Stefan Appel
Databases and Distributed Systems Group
TU Darmstadt

appel@dvs.tu-darmstadt.de
http://www.dvs.tu-darmstadt.de

Overview

TECHNISCHE UNIVERSITÄT DARMSTADT

- Event stream processing and business processes management
- Event stream processing units (SPUs): Encapsulation of event stream processing
- Integration of event stream processing:
  - Business process modeling
  - Business process execution
  - Enterprise IT infrastructure

Execution layer: Mapping from Model to Execution

TECHNISCHE UNIVERSITÄT DARMSTADT

- "Event compatible" equivalent to service invocation
- Explicit instantiation: Triggered when control flow reaches task

  `EsptInstantiate(MonitorShipment, MonitoringEvent, ShipmentId, 42)`
- Implicit instantiation: Data provided at registration of process model

  `EsptRegister(MonitorShipment, MonitoringEvent, ShipmentId)`
  - Automatic creation of SPU instances and synchronization with execution engine

Subscription example:

"Type==MonitoringEvent AND ShipmentID==42"