Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

"Model-Driven Performance Evaluation for Service Engineering"

Seminar Emerging Web Services Technology

David Jaeger

Agenda

1. Service- and Model-Driven Engineering
2. Performance Evaluation
3. Empirical Model-Driven Performance Evaluation
4. Monitoring
5. Evaluation Framework

Background Information

Dr. Claus Pahl
Dublin City University

Service and Software Engineering

Marko Boskovic
University of Oldenburg

Model-Driven Engineering, Performance Engineering

Prof. Dr. Wilhelm Hasselbring
University of Oldenburg

Software System Quality, Distributed Systems

- Published in Proceedings of *European Conference on Web Services* in November 2007

1. Service- and Model-Driven Engineering
2. Performance Evaluation
3. Empirical Model-Driven Performance Evaluation
4. Monitoring
5. Evaluation Framework

Service Engineering

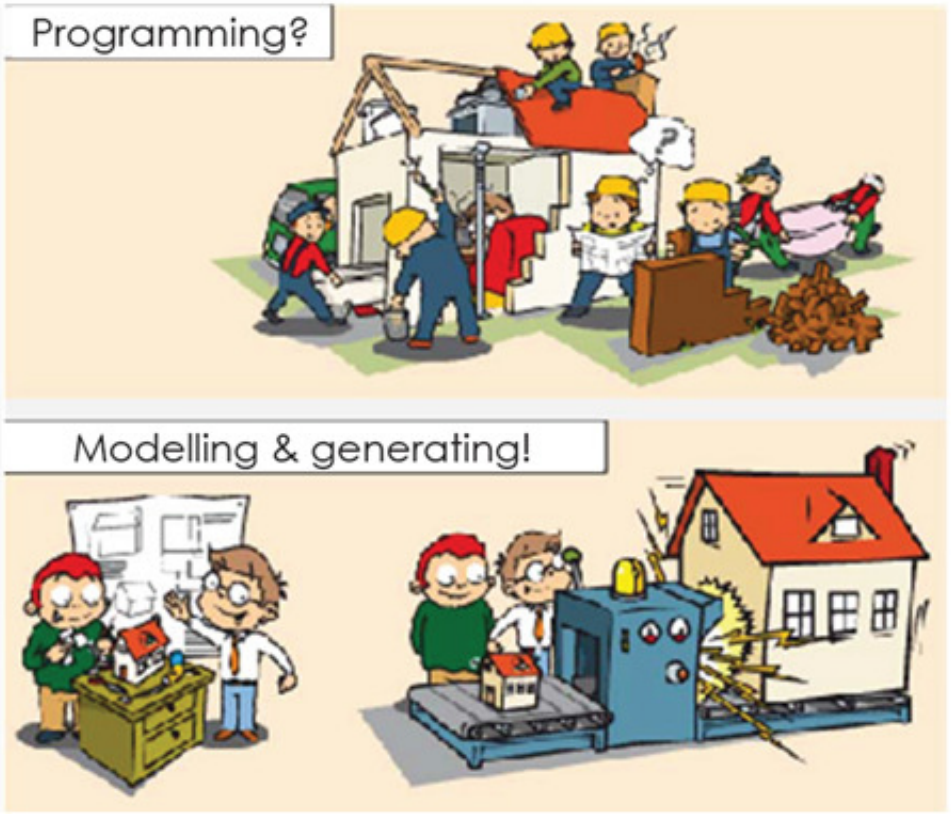
- Services are getting more **complex** over time
- **Composition** of services is major topic in research and business

- **Architectural questions** getting important
  - Hard to oversee all technologies and code
  - Do not cope with implementation details anymore

- Shift focus to **problem domain**
- Introduction of models as **abstraction**

Model-Driven Engineering (MDE)

- Key Points:
  - Discourage algorithmic and code concepts
  - Prefer Models as Abstraction
- Advantages
  - Formal analysis and evaluation of model
  - Generation of implementation from models
- Employment of Model-Driven Architecture (MDA)



[Metaphor by Johan den Haan]

Model-Driven Architecture (MDA)

- Popular MDE Approach by the *Object Management Group* (OMG)

- Guidelines
  1. Technologies => Problem domain
  2. Automation of relation between problem and implementation domain
  3. Open standards for interoperability

- Definition of models with domain-specific languages (DSL)
  - BPMN (Web Services)
  - UML

1. Service- and Model-Driven Engineering
2. Performance Evaluation
3. Empirical Model-Driven Performance Evaluation
4. Monitoring
5. Evaluation Framework

Performance and Quality of Service

- One of **Quality of Service** (QoS) attributes
  - Among reliability, availability and others
- Covered **metrics**
  - Response time
  - Throughput
  - Resource utilization

Motivation for Performance Evaluation

- Performance is **critical** property in today's business software
  - Demand for quality software
  - Client does not want to wait for long time (**timeliness**)
- Measurement of certain key properties
  - **Durations** in service composition
    - Single service action
    - End-to-End latency
  - **Responsiveness**
  - Number of **concurrent users**
  - **Resource consumption**

➢ Reveal **performance bottlenecks** and **improve service**

Problems with Performance Evaluation of SOAs

- Services are **deployed remotely**
  - No direct access
  - Cannot measure performance on one host
  - Measurement results must be collected from multiple locations
  - Network delay can influence performance
- Service **implementation** is probably **not available**
  - Neither as binary nor as code
  - Cannot easily inject performance measurement code
  - WSDL-file is only resource available

Evaluation Methods

Simulation

- Imitation of program execution focusing on certain aspect
- Pros: flexible
- Cons: Lack of accuracy

Analysis

- Mathematical description of system
- Pros: Easy to construct
- Cons: lack of accuracy (because of abstraction)

Empirical Evaluation

- Measurements and metrics calculation on real system
- Pros: Very accurate

1. Service- and Model-Driven Engineering
2. Performance Evaluation
3. Empirical Model-Driven Performance Evaluation
4. Monitoring
5. Evaluation Framework

Model-based empirical evaluation

- Evaluation approach chosen in paper
- **Model-based**
  - MDE fits the requirements of services
  - Empirical evaluation has already been researched on code-level
- **Empirical**
  - Accuracy benefits
  - Lacking research for model-level

1. Service- and Model-Driven Engineering
2. Performance Evaluation
3. Empirical Model-Driven Performance Evaluation
4. Monitoring
5. Evaluation Framework

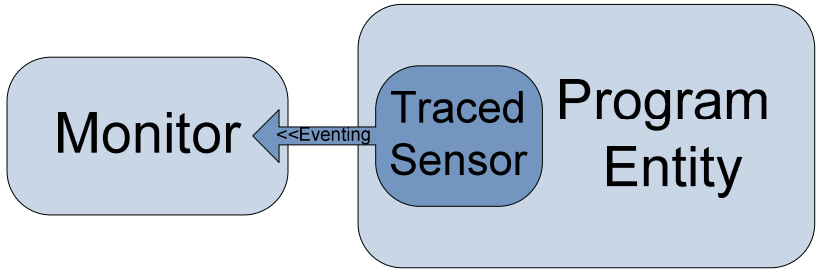
Sensors

- Monitoring is performed by means of **sensors**
  - Collect information about state of system

- Two types of sensors exist

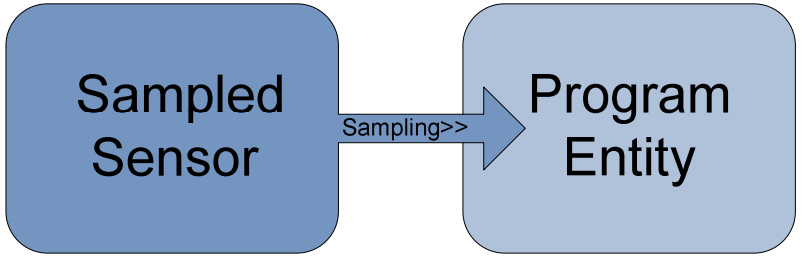
Traced

- Requires code in traced software
- Influences performance



Sampled

- Performance not influenced
- Infrequent state changes could be omitted

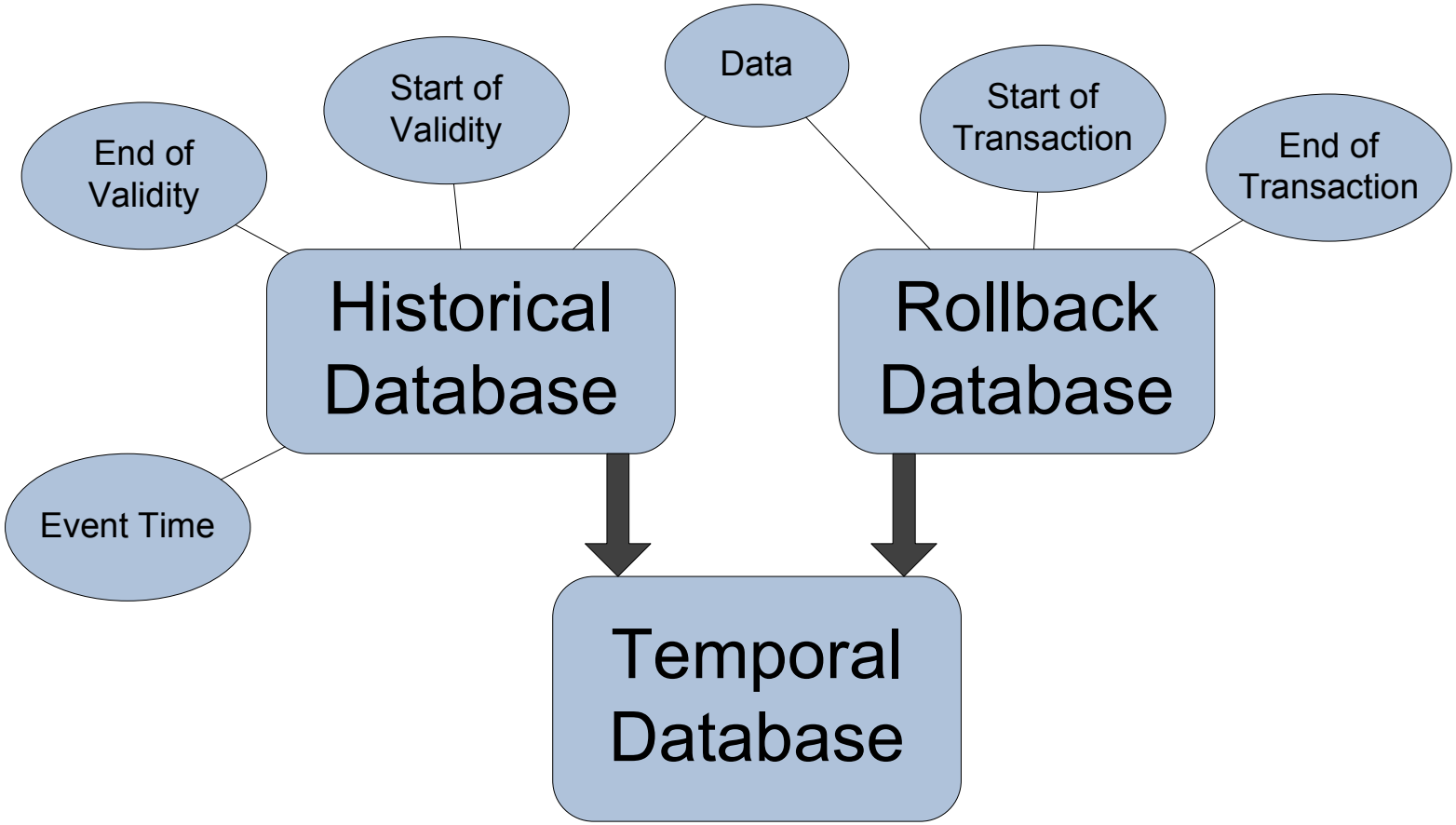


Recording Monitoring Data

- Recording of data emitted by sensors
  - Data: Time-varying relationship between entities of a computation
- Conventional **relational databases** are static
  - Record state at single moment of time
  - Current state of database is snapshot of system
- Extend relational databases
  - Record facts with corresponding time information

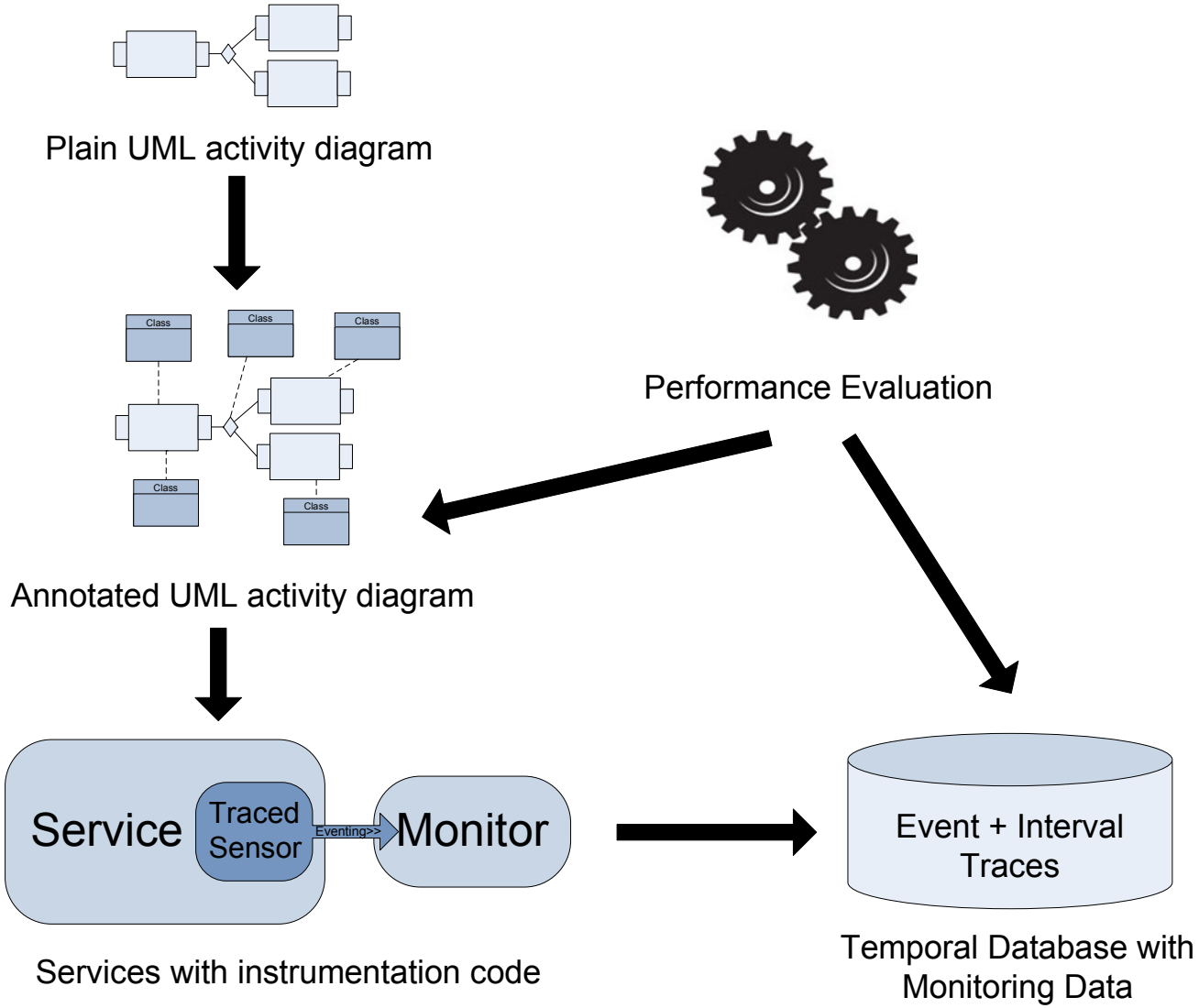
Temporal Databases

- Two distinct types of databases support recording of data with time information

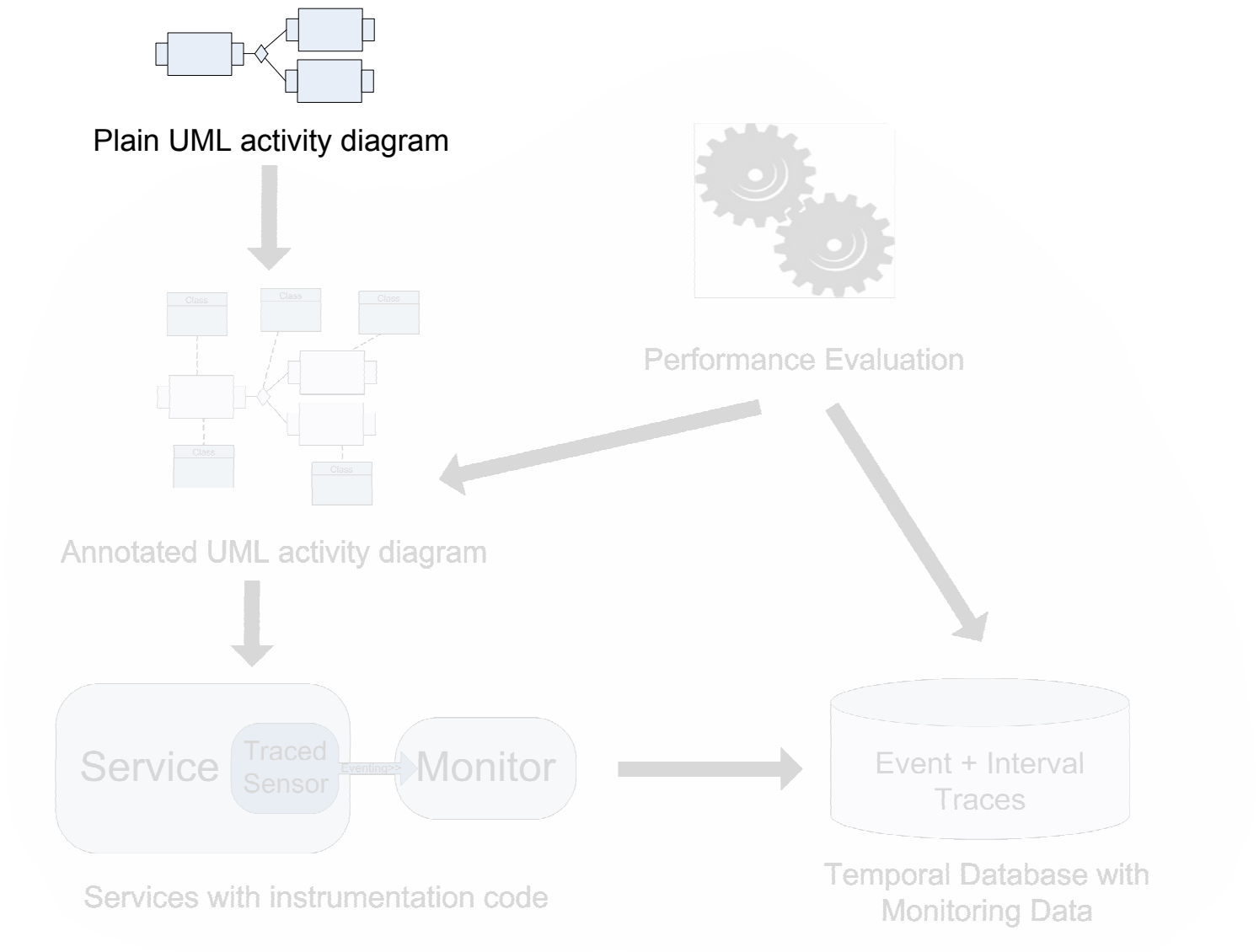


1. Service- and Model-Driven Engineering
2. Performance Evaluation
3. Empirical Model-Driven Performance Evaluation
4. Monitoring
5. Evaluation Framework

Overview of Framework Workflow

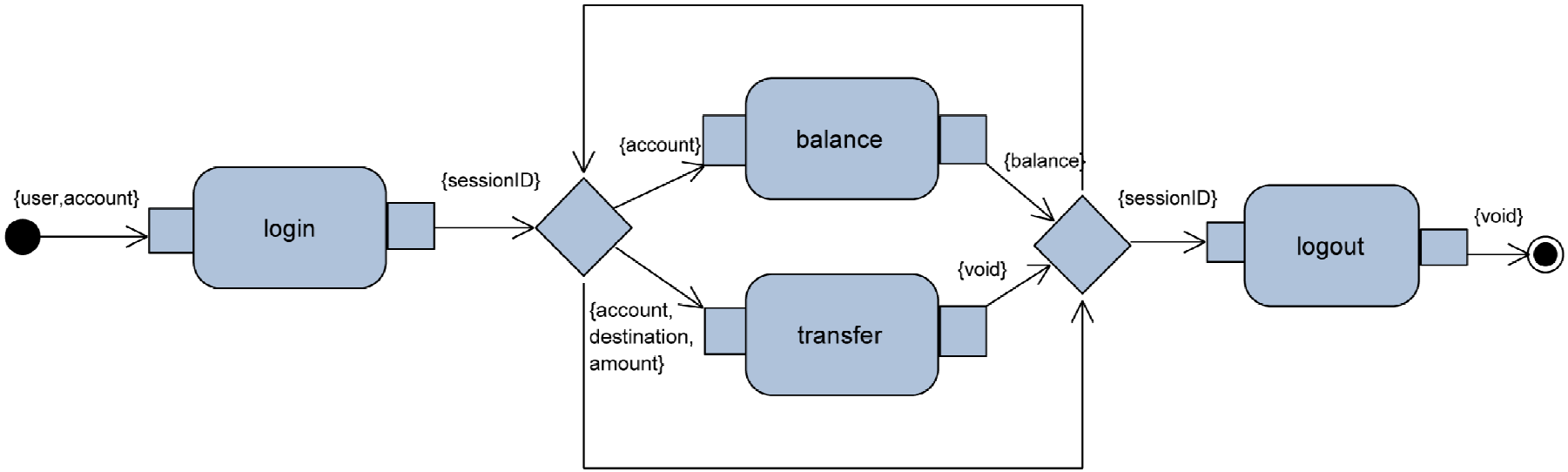


Step 1: Plain UML Activity Diagram

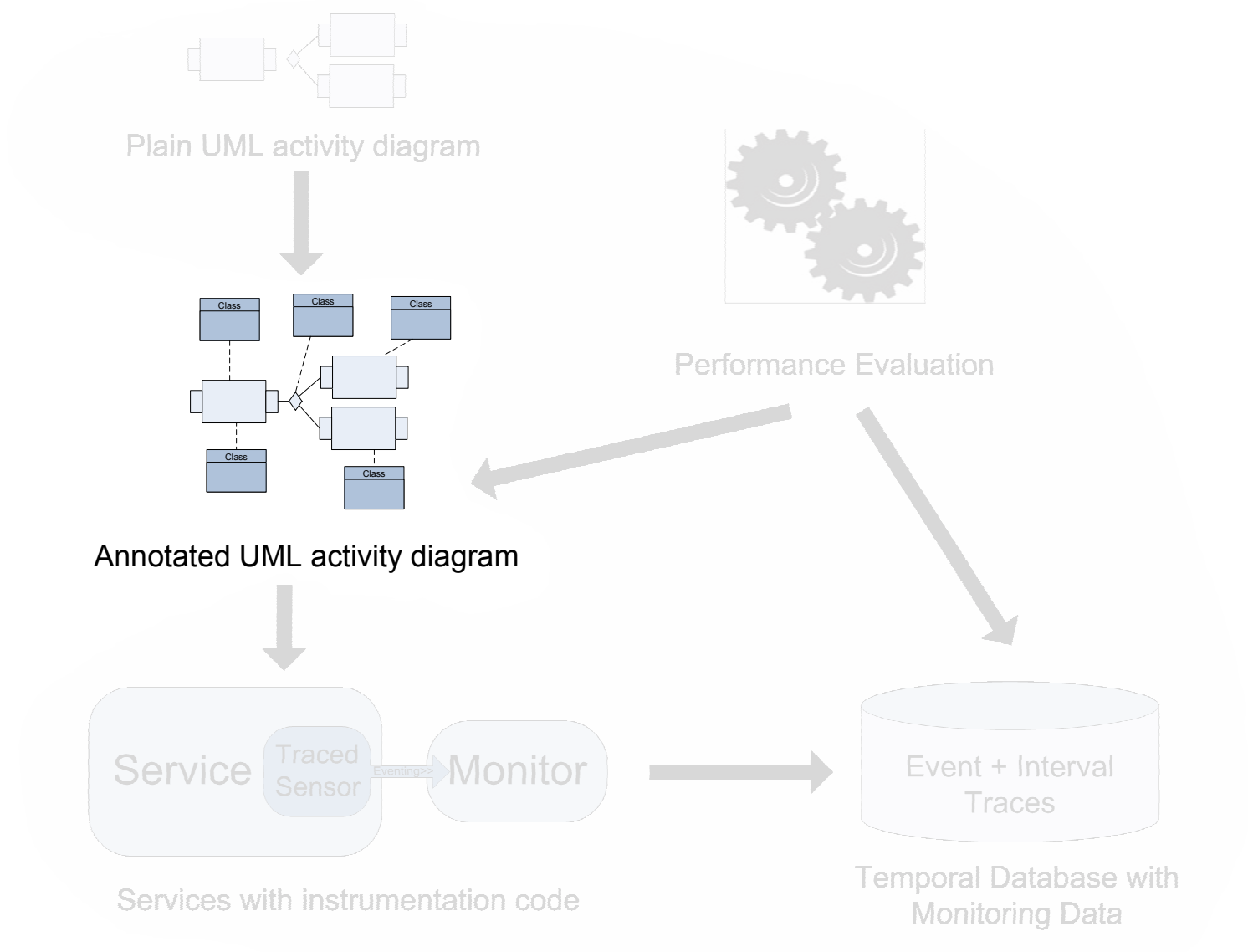


Plain UML Activity Diagram

- Model of the service process
  - Created by user/software designer
  - Modeled as UML activity diagram
    - Best fits requirements of extensibility

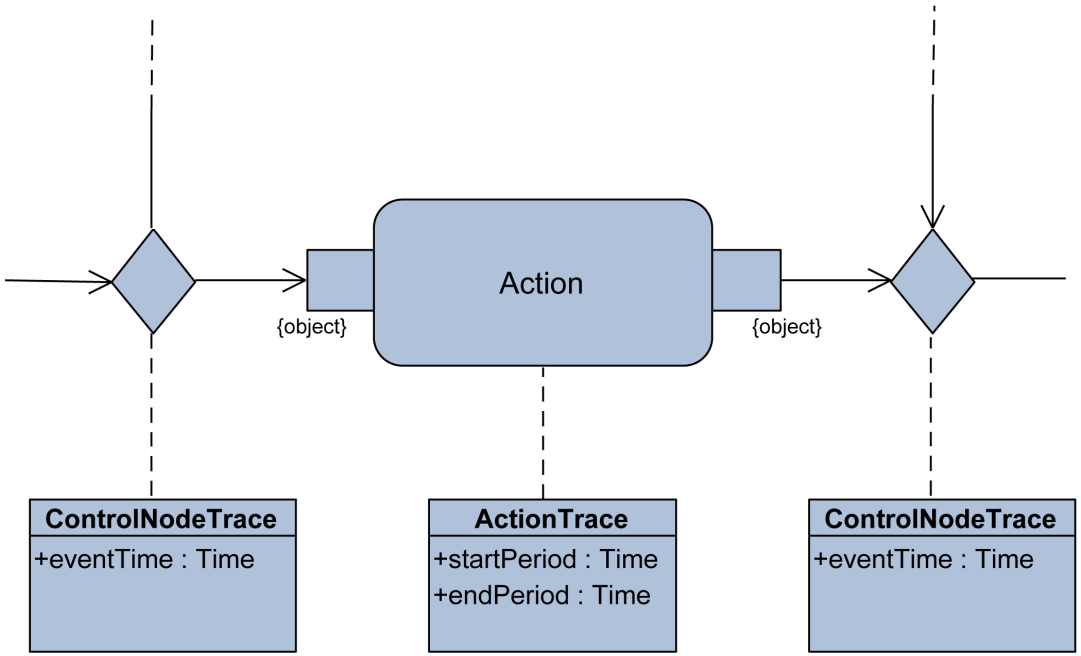


Step 2:
Monitoring Annotation for the Model



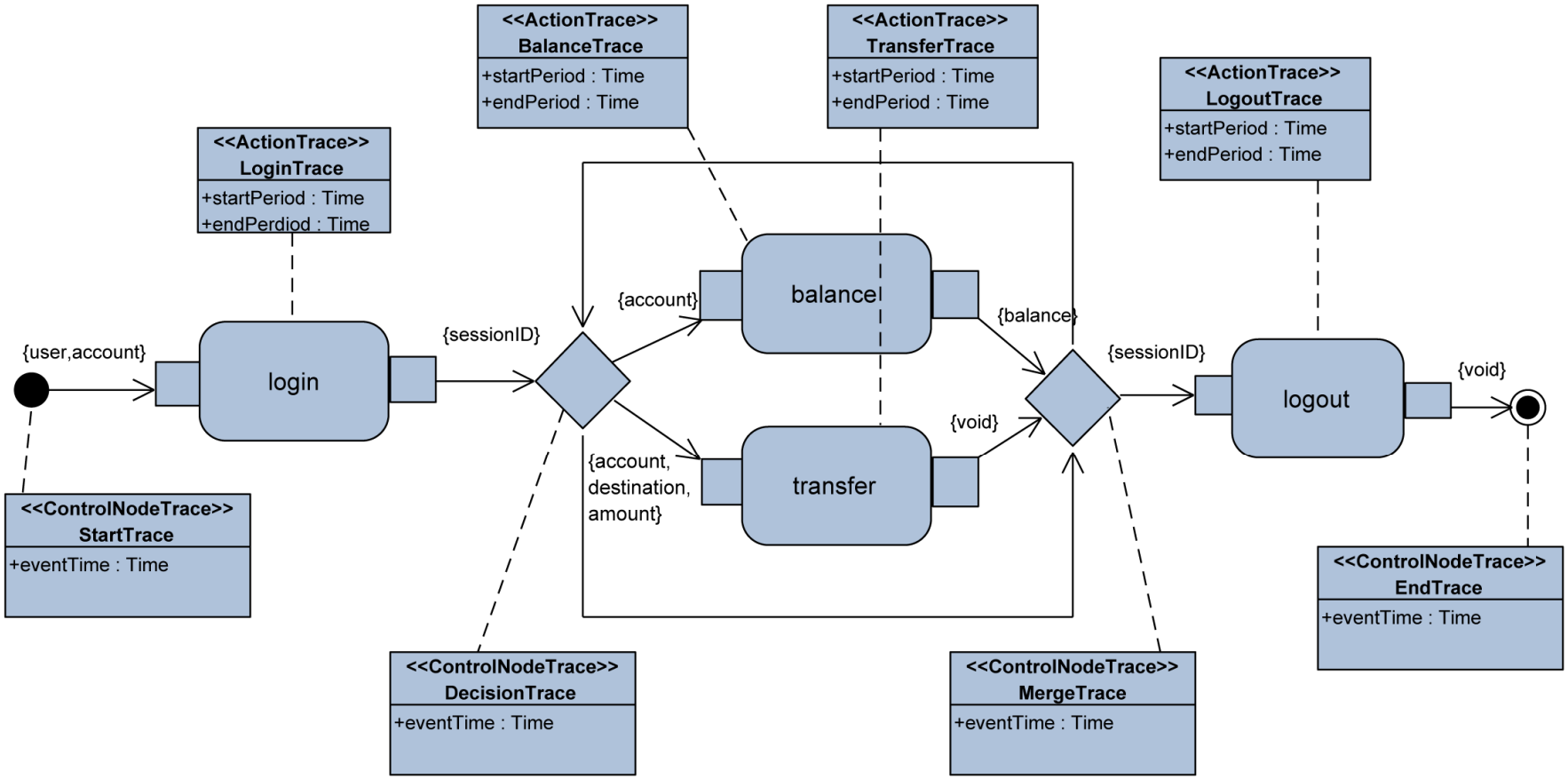
Annotation Entities

- Two types of annotations proposed
  - Each stands for certain trace type (Event, Interval)
- Events used for control nodes, Intervals used for action nodes

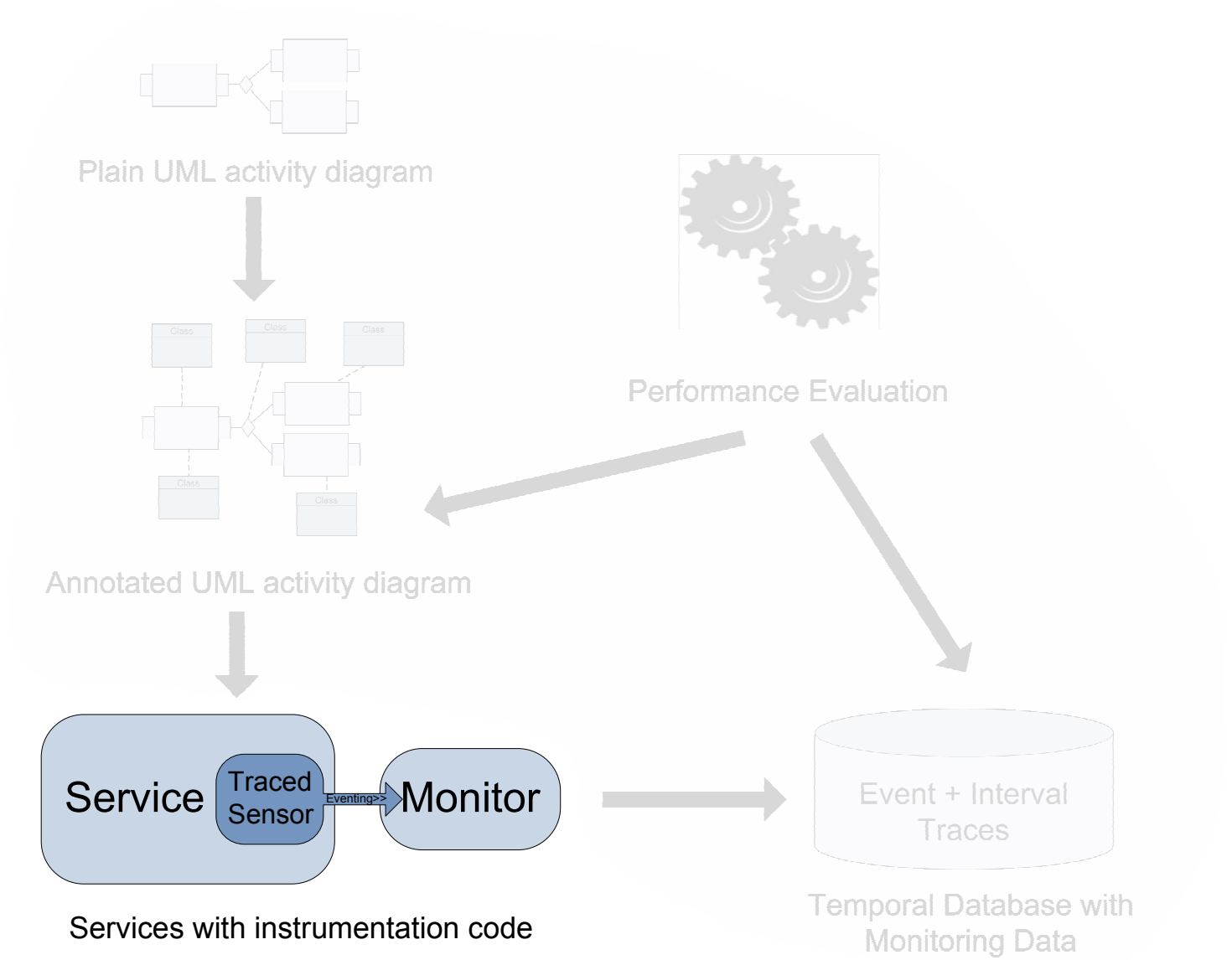


Monitoring Annotation for the Model

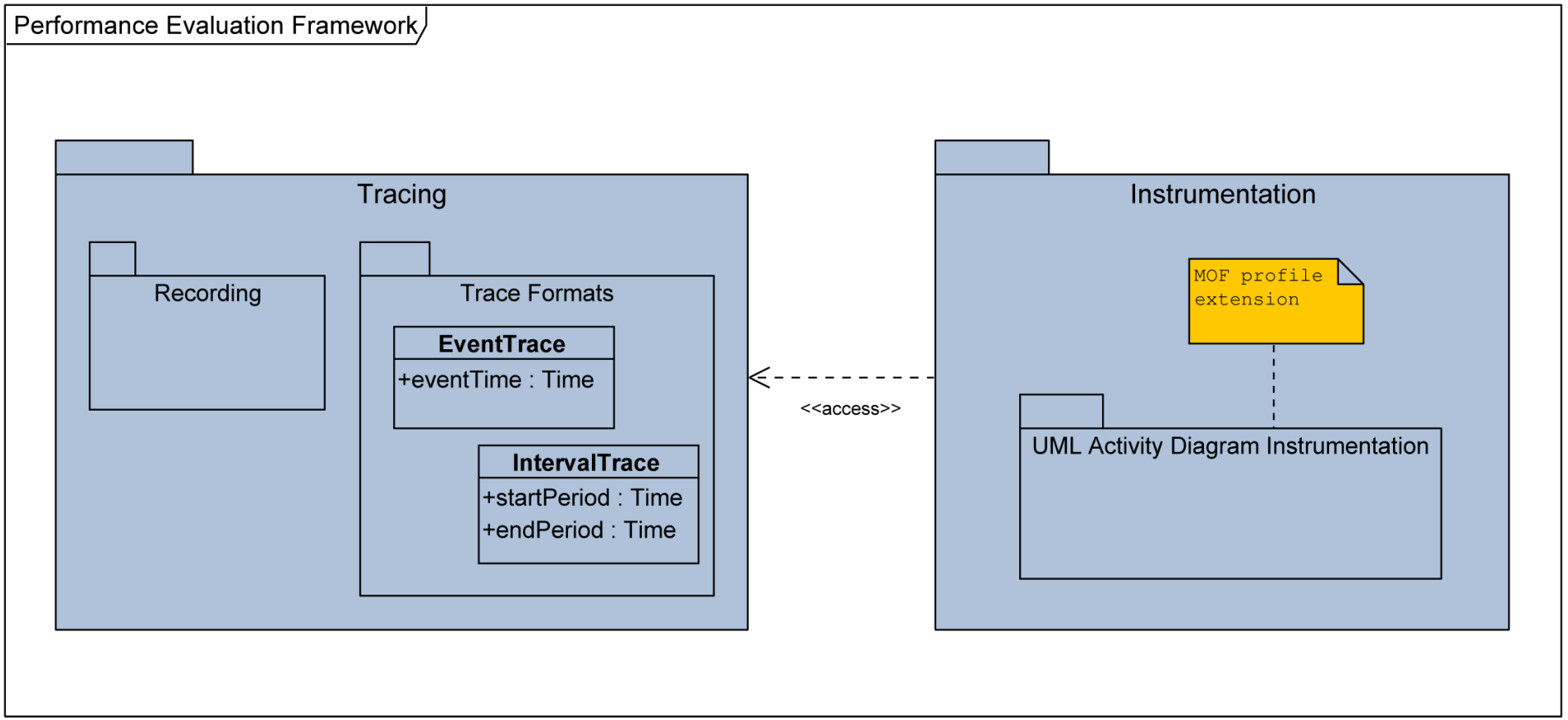
- Add annotations for instrumentation to plain model
  - Automatically or manually
- Each decision and action node gets corresponding trace annotation



Step 3: Instrumentation of the Code



Implementation: Package Structure



Tracing Package

- Actions
  - Intercepted at services
  - Collect start and end time of service
  - Send to temporal database
- Control nodes
  - Intercepted at process engine
  - Take single timestamp
  - Send to temporal database

Instrumentation of the Code

- Inject sensors into the services
  - Easy to realize
  - No significant performance overhead

Aspect Oriented Programming (AOP)

- Controlled environment with access to code
- Separation of instrumentation from code

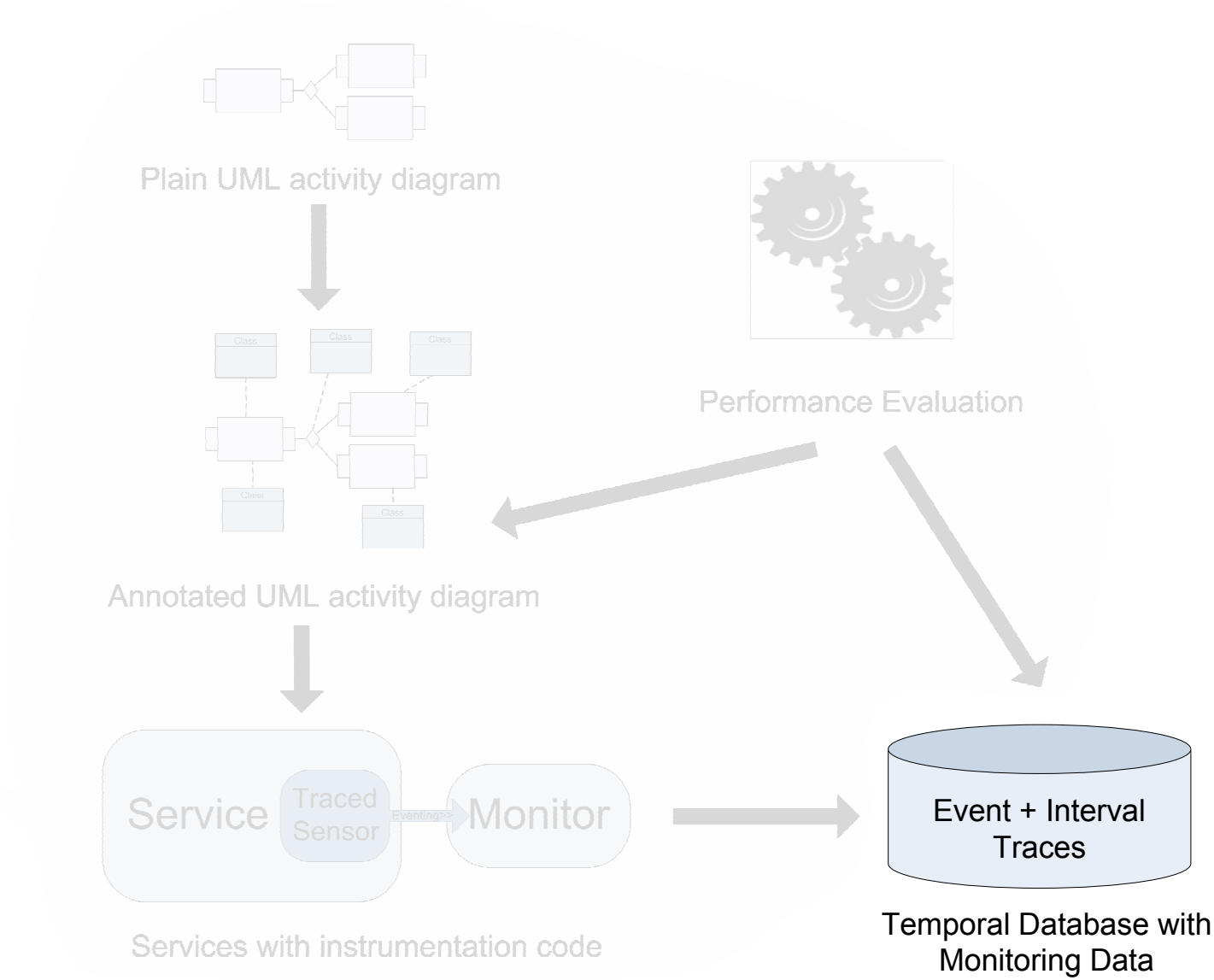
Interceptors

- Open environment with service black boxes
- Interception of method invocations with proxies

Generation of Code

- Instrumentation code generated automatically
- Employ ATLAS Transformation Language (ATL)
  - Input: UML activity diagrams with annotations
    - Service locations needed
  - Output: AOP-based code

Step 4: Temporal Database



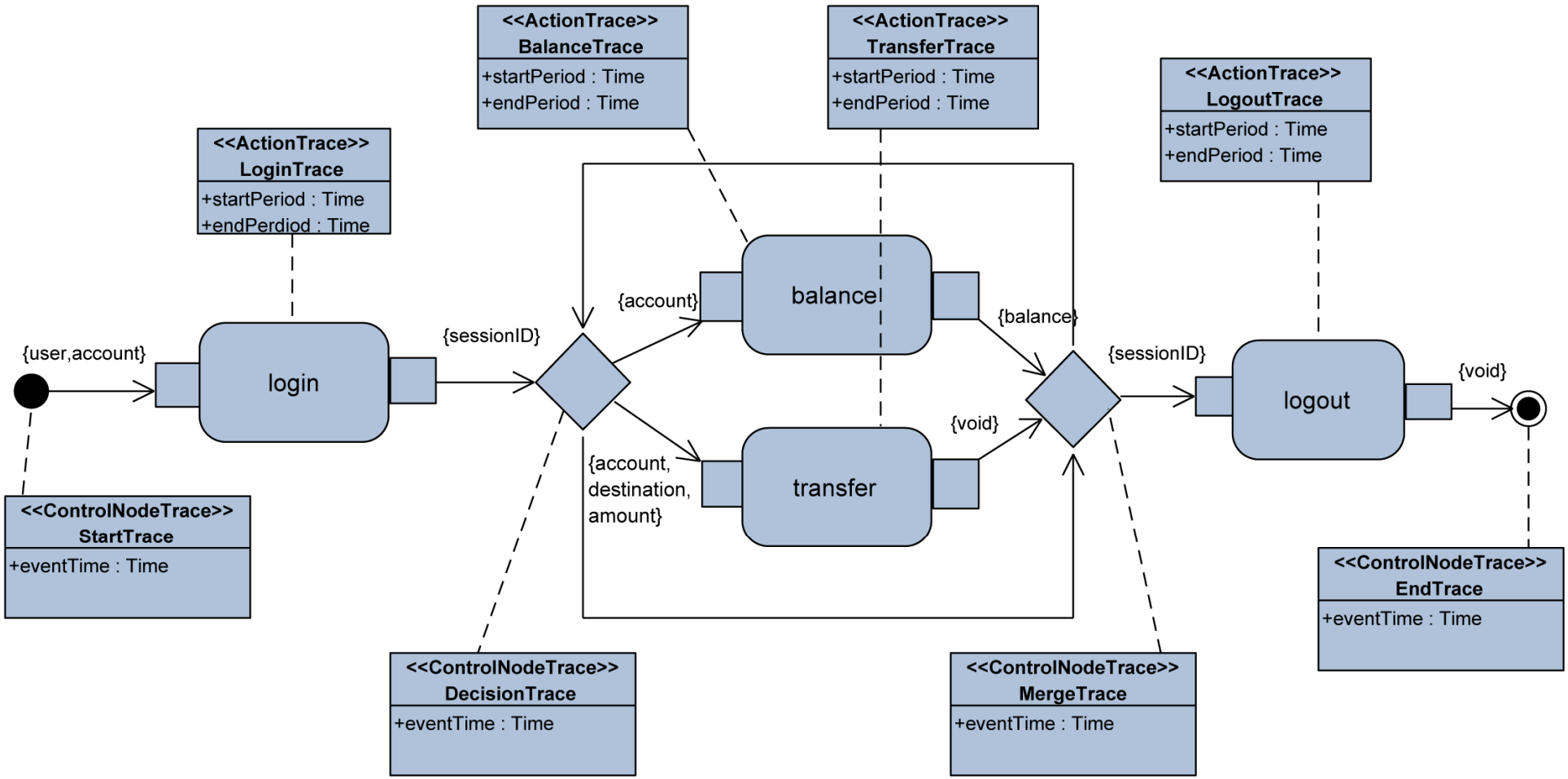
Temporal Database

- Two major implementations available
  - TimeDB
  - Oracle servers

- Database Structure
  - Single table for every sensor

TransferTrace	
startPeriod	**endPeriod**
2:22	2:45
3:03	3:12
3:15	3:29

DecisionTrace
eventTime
2:19
2:50
3:01
3:10


<<ActionTrace>>
BalanceTrace
+startPeriod : Time
+endPeriod : Time
<<ActionTrace>>
TransferTrace
+startPeriod : Time
+endPeriod : Time
<<ActionTrace>>
LogoutTrace
+startPeriod : Time
+endPeriod : Time
<<ActionTrace>>
LoginTrace
+startPeriod : Time
+endPerdiod : Time
{user,account}
login
{sessionID}
{account}
balance
{balance}
{sessionID}
logout
{void}
{void}
{account,
destination,
amount}
transfer
<<ControlNodeTrace>>
StartTrace
+eventTime : Time
<<ControlNodeTrace>>
EndTrace
+eventTime : Time
<<ControlNodeTrace>>
DecisionTrace
+eventTime : Time
<<ControlNodeTrace>>
MergeTrace
+eventTime : Time

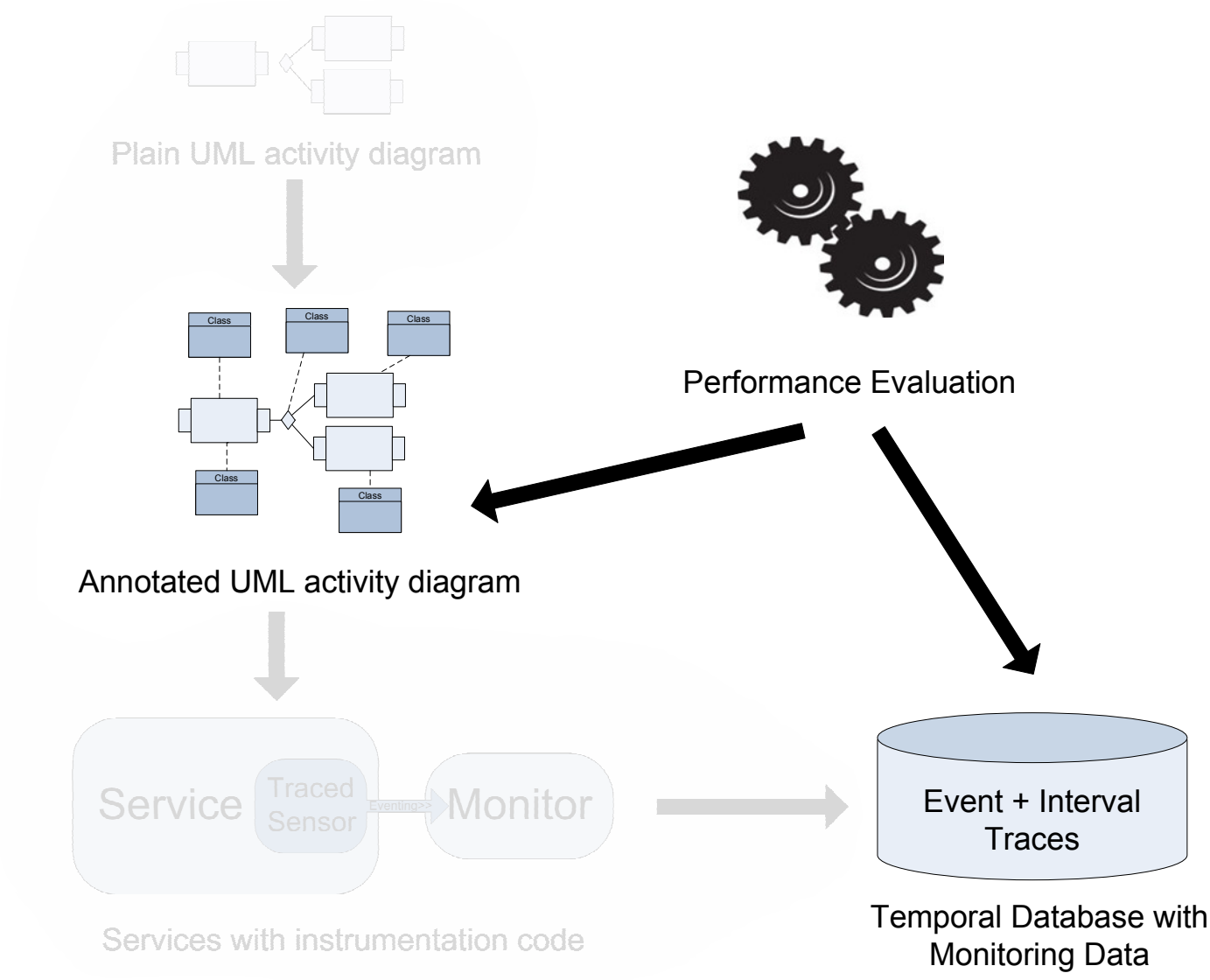
Temporal Database

- Two major implementations available
  - TimeDB
  - Oracle servers
- Database Structure
  - Single table for sensor type

ActionTraces		
type	**startPeriod**	**endPeriod**
login	2:22	2:45
balance	2:47	2:50
logout	2:52	2:54

ControlNodeTraces	
type	**eventTime**
start	2:21
decision	2:46
merge	2:51
end	2:55

Step 5: Evaluation of Results



Evaluation of Monitoring Data

- Can perform performance queries on temporal database
- Special query language required (TSQL2, TQuel)

- Evaluate response time of single service

```
SELECT CAST(VALID(AT) TO INTERVAL SECOND) / COUNT(AT.type)
FROM ActionTraces(type) AS AT
WHERE AT.type = 'balance'
```

- Evaluate the frequency of called services

```
SELECT COUNT(AT.type) / COUNT(CNT.type)
FROM ActionTraces(type) AS AT, ControlNodeTraces(type) AS CNT
WHERE AT.type = 'balance' AND CNT.type = 'decision'
```

Conclusion

- New approach for performance evaluation of Web Services
  - Focus on abstract model-layer
  - Evaluation by empirical analysis
- Good overview of time spent in single action and relations between certain control points
  - However…
    - Cannot associate measuring results of same walkthrough
    - No association between control points and actions
- No further work on the topic

Literature

[1] Pahl, C.; Boskovic, M.; Hasselbring, W.: *Model-Driven Performance Evaluation for Service Engineering*, 2007

[2] Snodgrass, R.: *A Relational Approach to Monitoring Complex Systems*, 1988

[3] Pahl, C. et alii: *Quality-Aware Model-Driven Service Engineering* in *Model Driven Software Development: Integrating Quality Assurance*, 2009

[4] Debusmann, M. et alii: *Measuring End-to-End Performance of CORBA Applications using a Generic Instrumentation Approach*, 2002

[5] Liao, Y.; Cohen, D.: *A Specificational Approach to High Level Program Monitoring and Measuring*, 1992

[6] Snodgrass, R.: *The TSQL2 temporal query language*, 1995

Questions?