SLA-basiertes Systemmanagement mit dem Common Information Model

Markus Debusmann
FH Wiesbaden, FB Informatik
Labor für Verteilte Systeme
Kurt-Schumacher-Ring 18
65197 Wiesbaden
debusmann@informatik.fh-wiesbaden.de

Alexander Keller
IBM Research Division
T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA
alexk@us.ibm.com

Inhalt

- Motivation
- Technologien
  - WSLA-Framework
  - WBEM/CIM
- Konzept
  - Integrationsalternativen & Informationsmodell
  - Architektur für SLA-basiertes Management
  - Aktive Provider
  - Laufzeitbeziehungen
- Implementierung
- Fazit

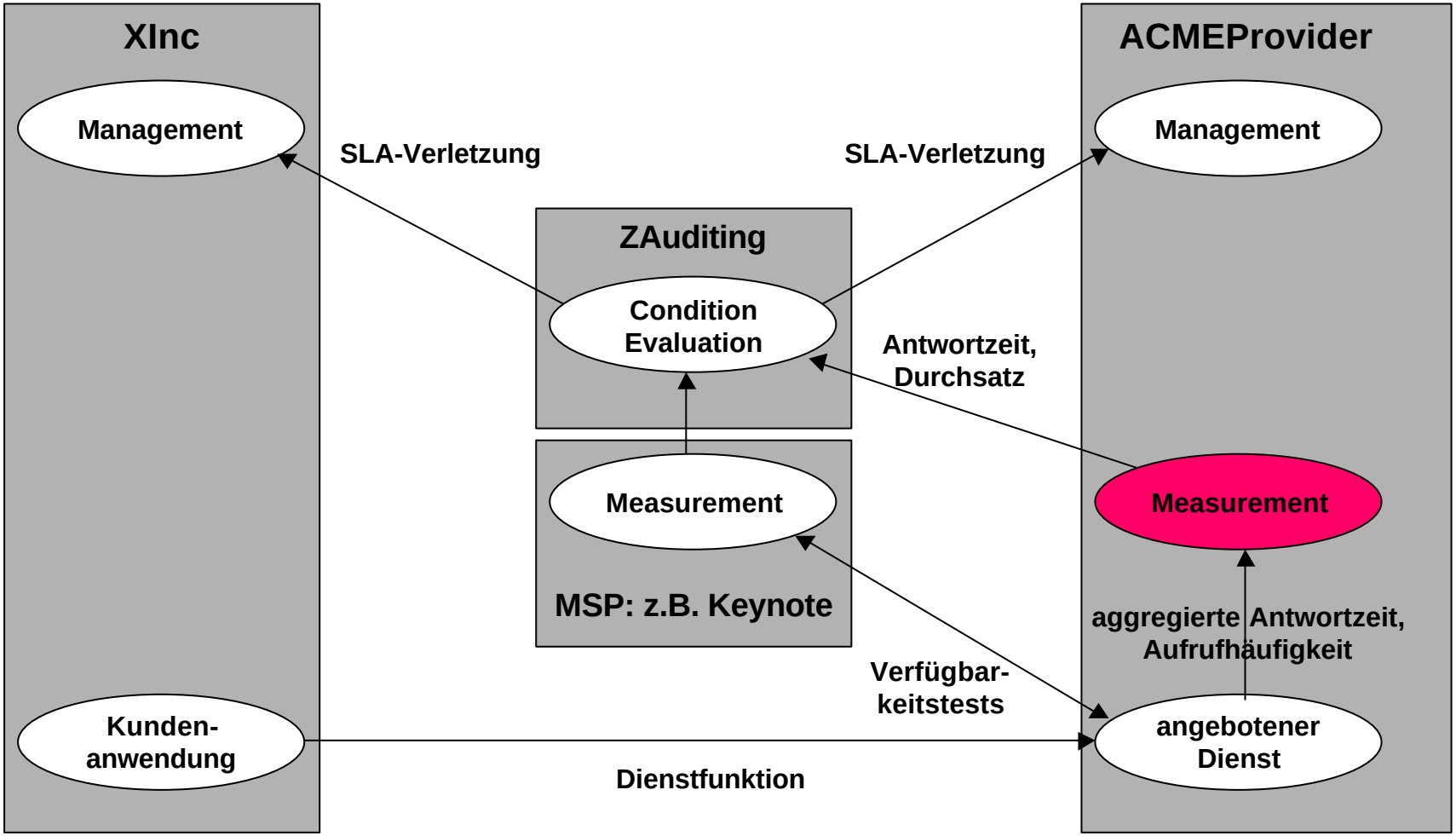
Motivation

- Anwendungsarchitekturen
  - Web Service orientierte Integrationsansätze sind von steigender Bedeutung
  - Open Grid Services Architecture als Basis von E-Business-Anwendungen
- Managementanforderungen
  - "On-demand" Bereitstellung von Diensten
  - dynamische Aushandlung und Überwachung von Service Level Agreements

Web Service Level Agreements (WSLA)

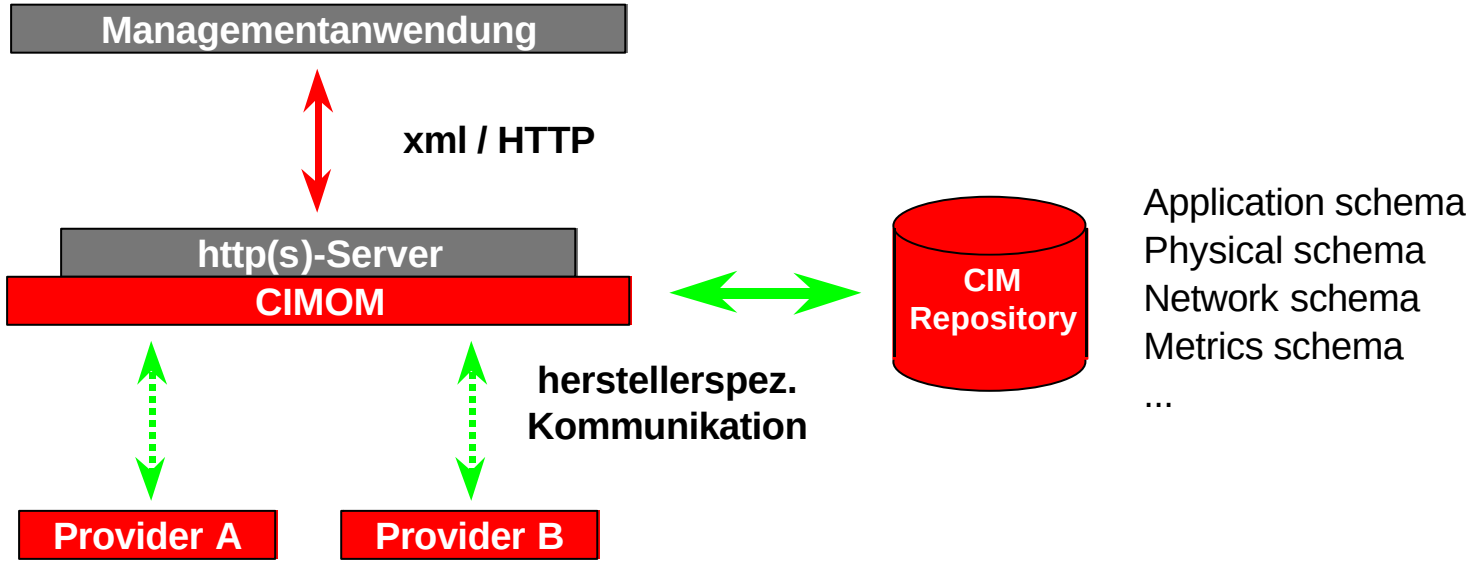
- WSLA von IBM Research entwickelt und Teil des IBM Web Services Toolkit 3.2
  - `http://www.alphaworks.ibm.com/tech/webservicetoolkit`
- SLA-Überwachung in Inter-Domain-Umgebungen
- Rollen
  - Dienstnutzer
  - Dienstanbieter
  - Drittanbieter
- Laufzeitumgebung
  - Deployment Service
  - Measurement Service
  - Condition Evaluation Service
- XML Schema zur Definition von SLAs

WSLA Dienste zur Laufzeit



WBEM / CIM

- **Web-based Enterprise Management**
  - **Integration existierender Managementansätze**
  - **Nutzt das Common Information Model zur Beschreibung von Ressourcen**
- **Common Information Model**
  - **Objektorientiertes Informationsmodell**
  - **Schemata definieren Basisklassen verschiedener Ressourcenkategorien**



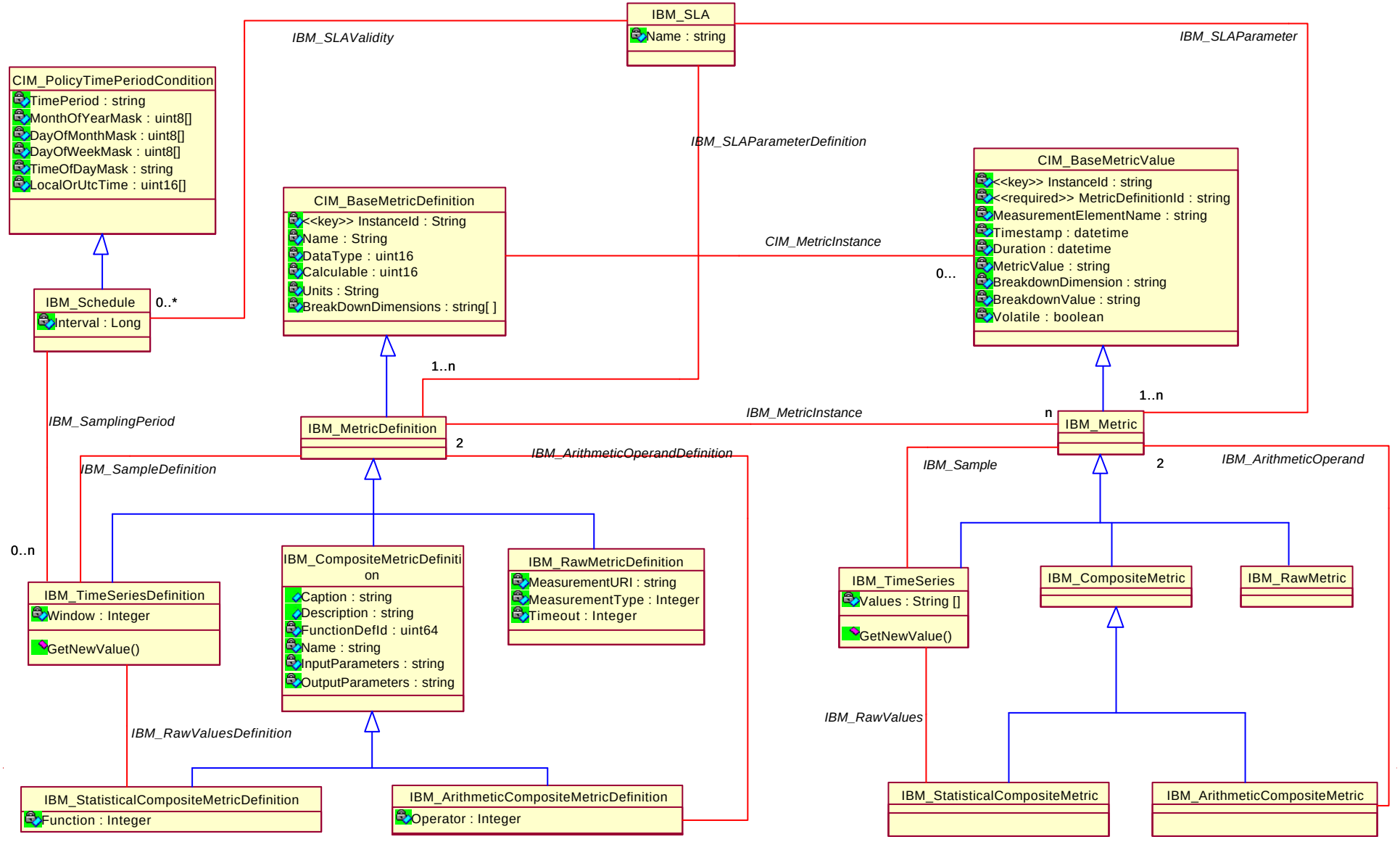
Integrationsansatz

- Einschränkung
  - Integration von WSLA und CIM nur auf der Seite des Dienstanbieters
- CIM-basierter Measurement Service
  - einfache Integration
    - in bestehende Managementumgebungen
    - von verfügbaren Ressourcen
- Condition Evaluation Service
  - weiterhin Web Service
  - einfache Auslagerung an Drittanbieter

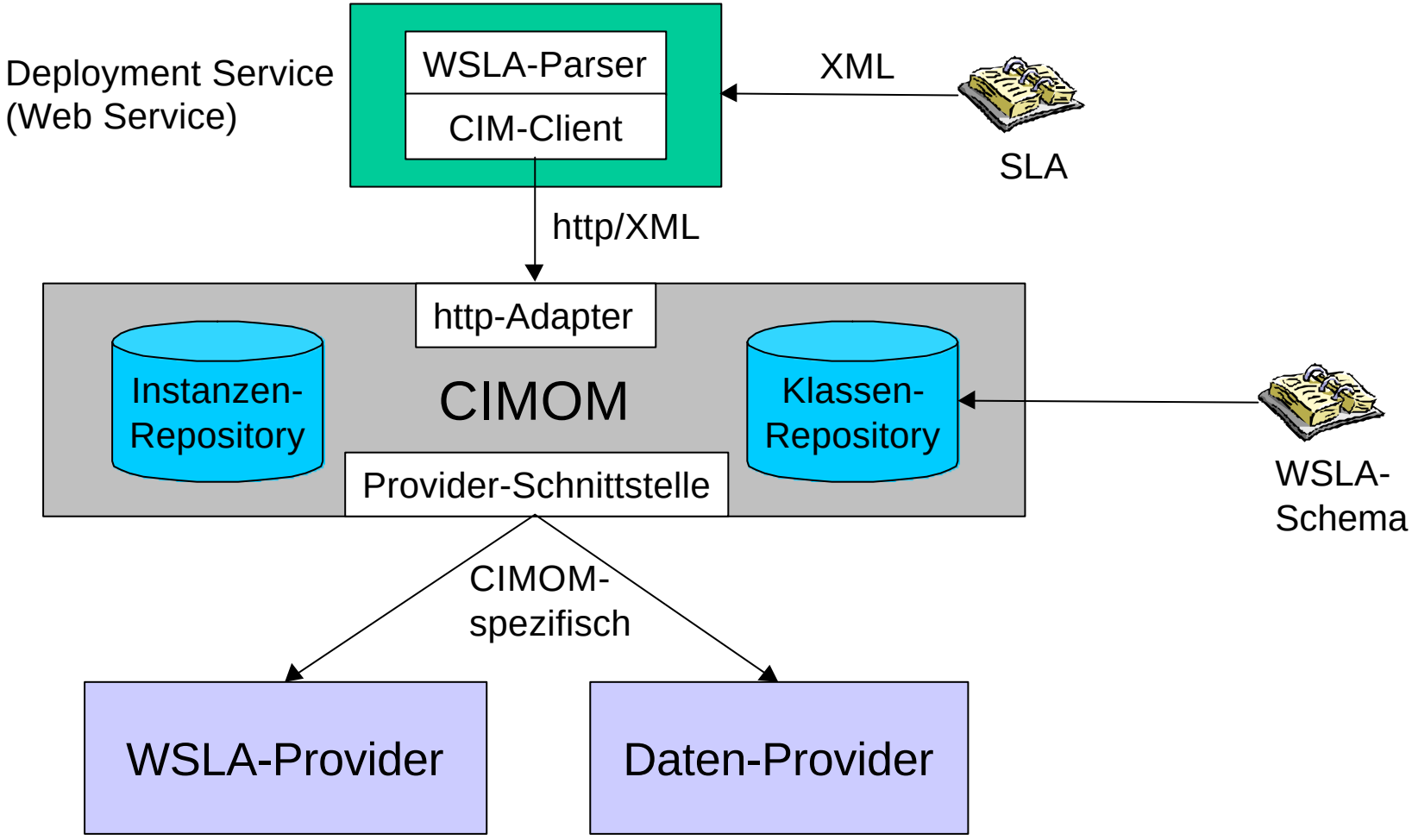
CIM-Informationsmodell

- Informationsmodell muss in WSLA definierte SLAs abbilden
- Beschränkung auf Measurement Service bedeutet
  - Beschreibung der Beziehungen zwischen SLA-Parametern und feingranularen Ressourcenmetriken
  - Definition von Berechnungsfunktionen
  - Schedules zum Holen und Berechnen von Metriken
- Metrikdefinitionen und Metrikinstanzen (analog CIM Metrics Schema)
  - Reduktion von Redundanz
  - Wiederverwendung von Metrikdefintionen ➔ Katalog von Metrikdefinitionen

Informationsmodell



Architektur des Measurement Service

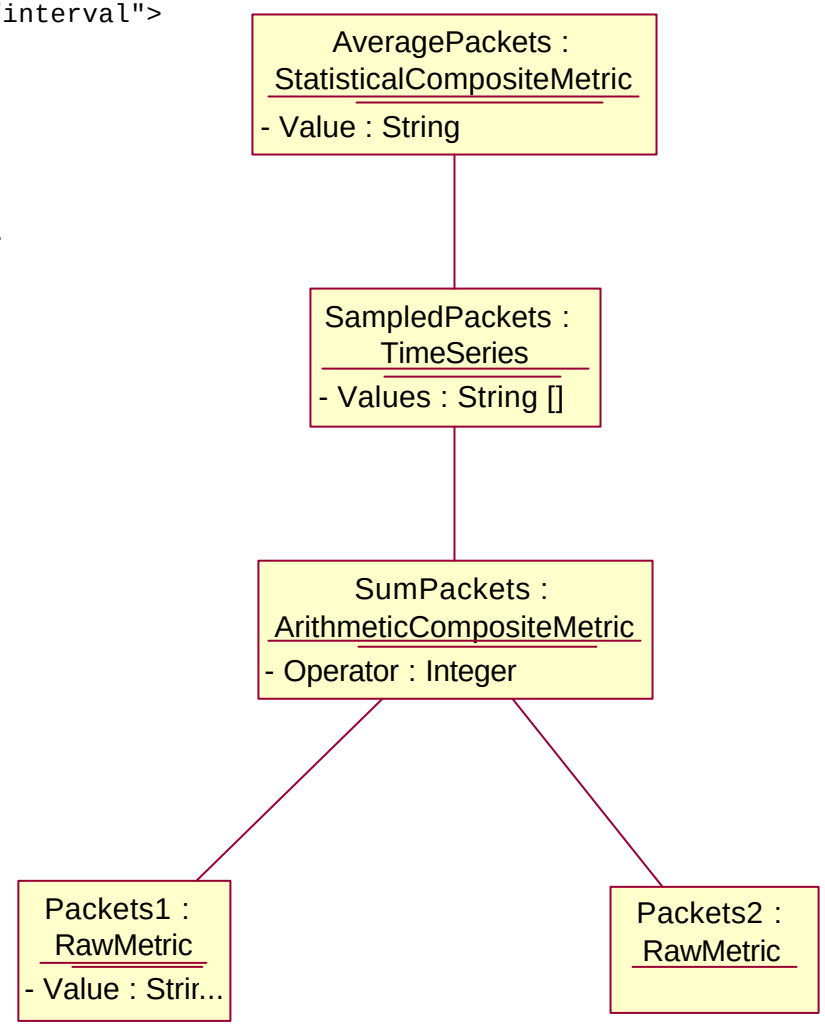


Aktive CIM-Provider

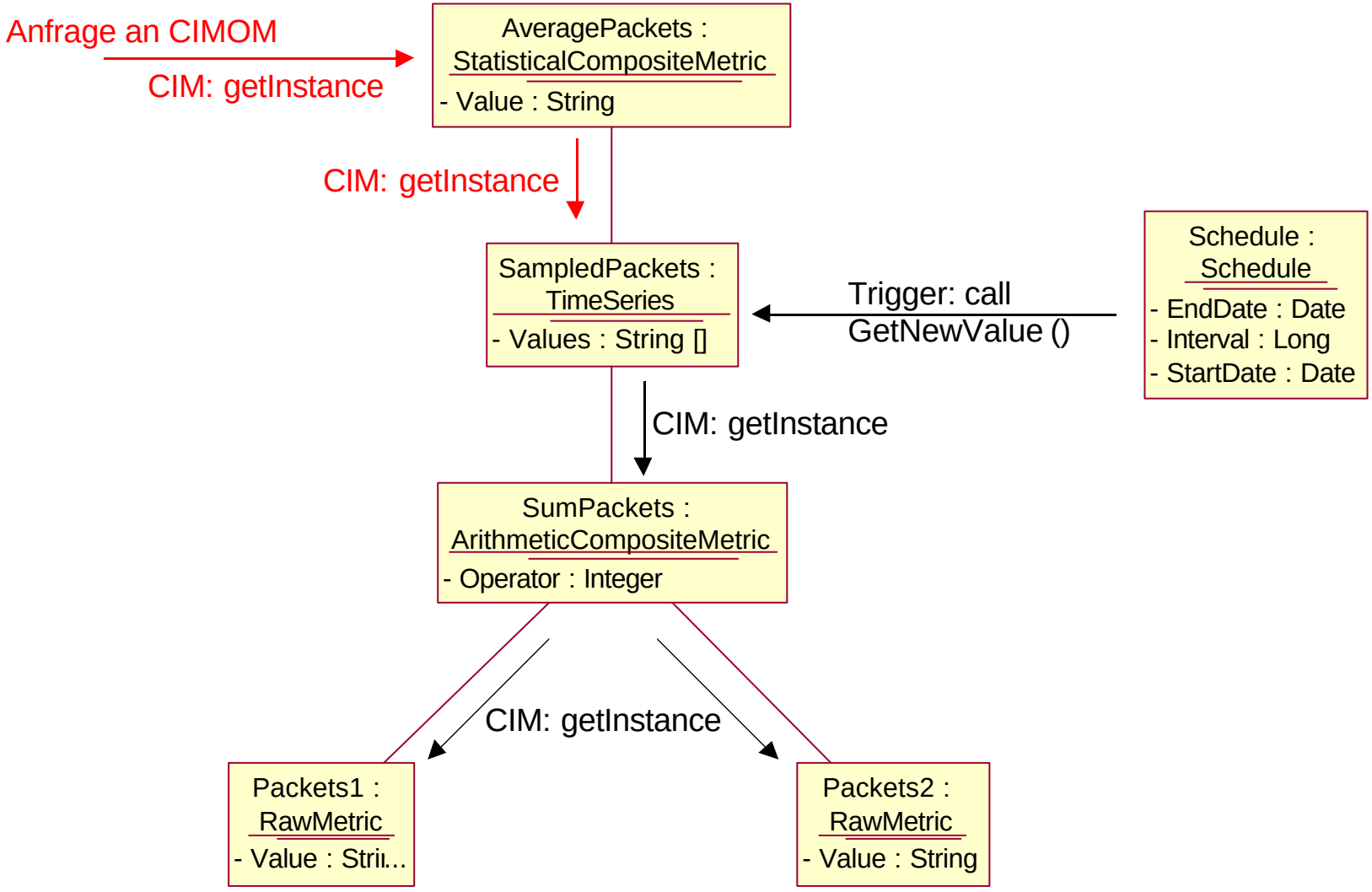
- **Bisher**
  - **Abfrage von Ressourcemetriken wird von Managementanwendung ausgelöst**
  - **"dumme" und passive Ressourcen-Provider**
  - **Kapseln von Legacy-Komponenten**
- **Jetzt**
  - **Abfrage der Ressourcen vom Provider selbständig initiiert (Klasse Schedule)**
  - **Persistenzmechanismus des CIMOMs wird für historische Daten genutzt**
  - **Geringerer Overhead**

Struktur eines einfachen SLA

```
<Metric name="AveragePackets" type="float" unit="packets/interval">
    <Source>YMeasurements</Source>
    <Function resultType="float" xsi:type="Average">
       <Metric>SampledPackets</Metric>
    </Function>
  </Metric>

  <Metric name="SampledPackets" type="TS" unit="packets">
    <Source>YMeasurements</Source>
    <Function resultType="TS" xsi:type="TSConstructor">
      <Schedule>availabilityschedule</Schedule>
      <Metric>SumPackets</Metric>
      <Window>4</Window>
    </Function>
  </Metric>

<Metric name="SumPackets" type="float" unit="packets">
  <Source>YMeasurements</Source>
  <Function resultType="float" xsi:type="Plus">
    <Operand>  <Metric> Packets1 </Metric>  </Operand>
    <Operand>  <Metric> Packets2 </Metric>  </Operand>
  </Function>
</Metric>

<Metric name="Packets1" type="float" unit="packets">
  <Source>YMeasurements</Source>
  <MeasurementDirective resultType="integer
                        xsi:type="InvocationCount">
    <CounterURI>http://...</CounterURI>
  </MeasurementDirective>
</Metric>

<Metric name="Packets2" type="float" unit="packets">
...
</Metric>
```



Kontrollflüsse zur Laufzeit



Prototypische Implementierung

- Implementiert in Java und dem SNIA CIMOM
- Prinzipiell wird jede CIM-Klasse durch einen Provider bereitgestellt
- Implementierungsaufwand minimiert durch wiederverwendbare Funktionalität in Basisklassen
- Komplexere Provider für
  - Schedule-Klasse
  - Metrikinstanzen

Das "CIM-Bootstrapping"-Problem

- **Problem:**
  - **CIM Provider werden bei der ersten Anfrage geladen**
- **Deployment eines SLAs**
  - **CIMOM muss notwendige Provider laden**
    → aktives Monitoring beginnt
- **Neustart des CIMOM**
  - **Provider werden nicht automatisch geladen**
    → bisherige SLAs werden nicht überwacht
- **Lösung:**
  - **Enumerieren der Schedule-Instanzen nach dem Neustart des CIMOM**
    → notwendige Provider werden geladen und SLAs werden überwacht

Fazit

- **Prototyp zeigt die prinzipielle Integration von WSLA und CIM**
  - **WSLA-Modell kann auf CIM abgebildet werden**
  - **Measurement Service ist als CIM-Provider implementierbar**
- **Aktives Monitoring ist Neuland in CIM**
- **Spezielle Bootstrapping-Prozedur notwendig beim Neustart des CIMOMs**