

# **SLA Monitoring:** The SLA@SOI Monitoring Architecture

George Spanoudakis  
CITY UNIVERSITY London

Date: 28/9/2010

**Motivation & Required Capabilities**

**The SLA@SOI Monitoring Architecture**

**Monitorability Check & Dynamic Set up**

**SLA Translation**

**Overview of Key Innovations**

**Open Issues & Ongoing Work**

## Motivation & Required Capabilities

## The SLA@SOI Monitoring Architecture

## Monitorability Check & Dynamic Set up

## SLA Translation

## Overview of Key Innovations

## Open Issues & Ongoing Work

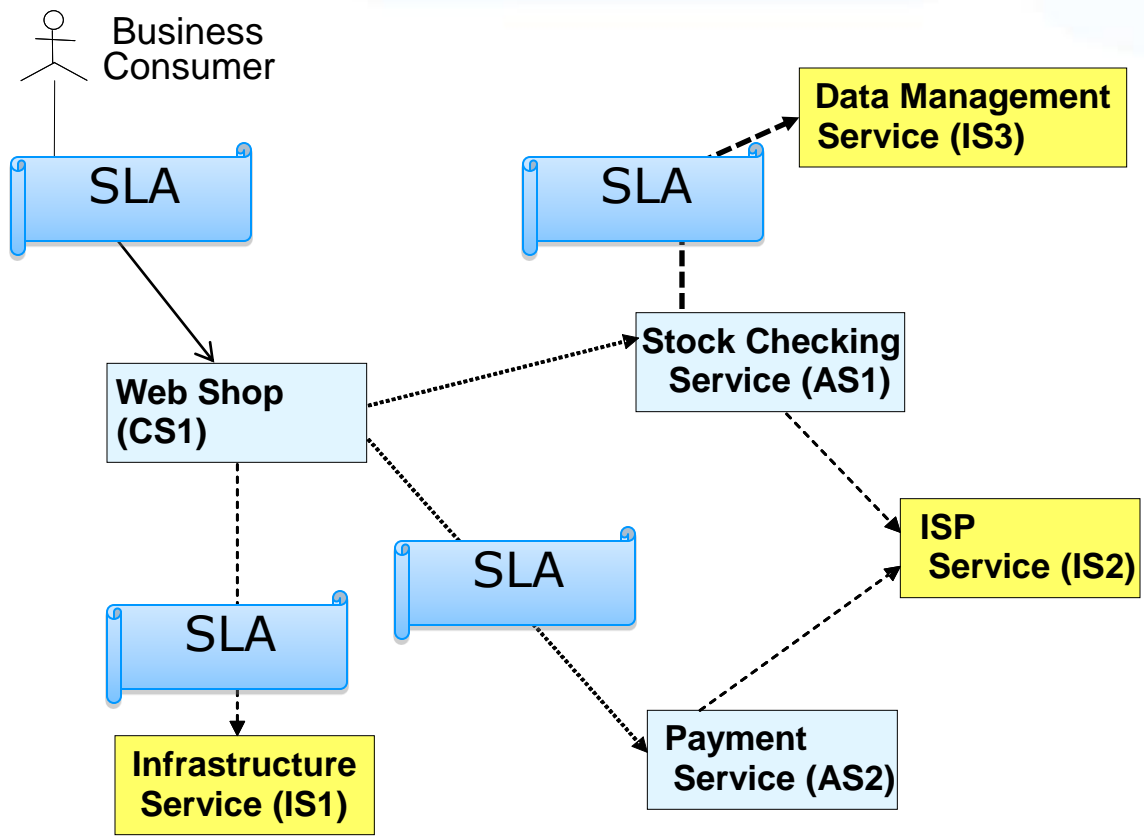
## Complex service based systems

- Deploy atomic and composite software services that may change dynamically
- Run on infrastructures (available as infrastructure services) that may change dynamically
- Are regulated by Service Level Agreements (SLAs) that may change dynamically



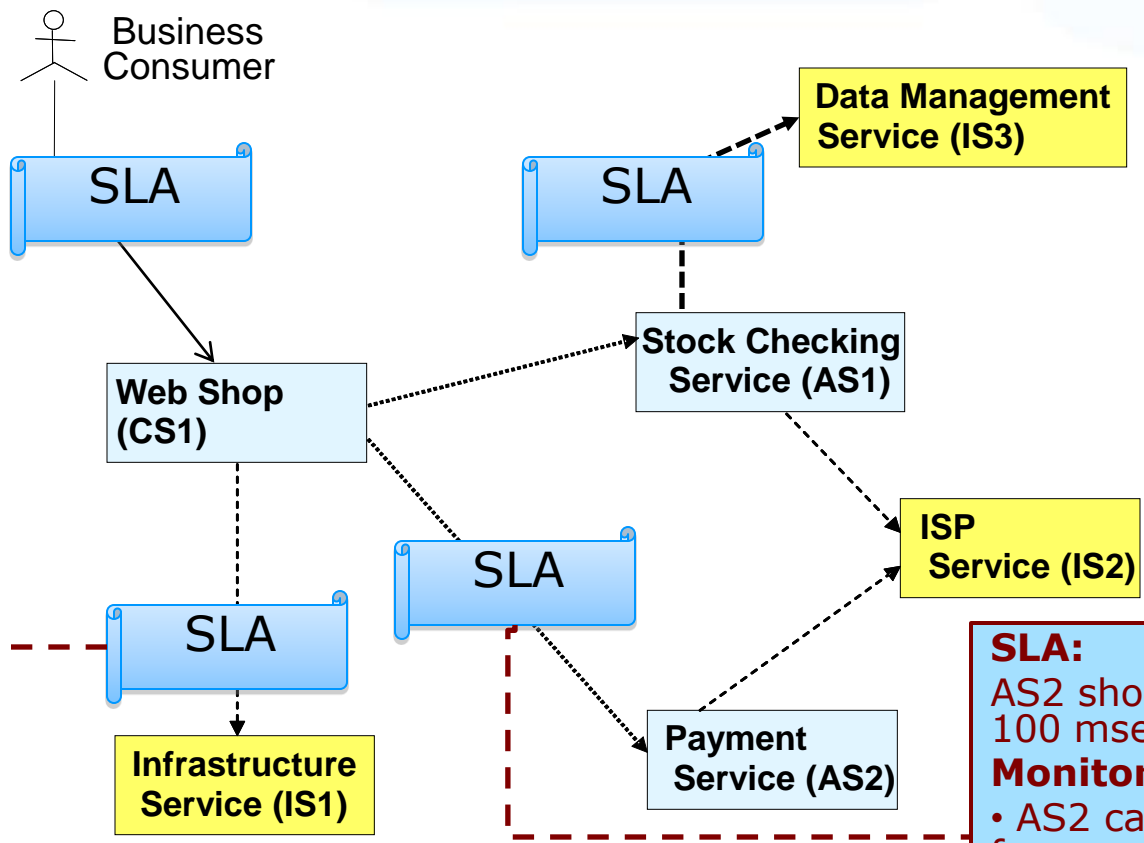
Effective SLA management requires  
**heterogeneous** and **dynamically configurable** and  
**adaptable** infrastructures for **SLA monitoring**

# Example: Web shop Service



- > deployed on/deploys
- > composed of
- > uses

# Example: Web Shop Service



- - - - -> deployed on/deloys  
 ———> composed of  
 ———> uses

**SLA:**  
 Server IS1 throughput should be X requests/msec  
**Monitoring:**  
 Server's throughput can be obtained by local system calls

**SLA:**  
 AS2 should produce a response in less than 100 msec on average  
**Monitoring:**

- AS2 can provide **events** capturing calls from and responses to CS1 to enable CS1 monitor its availability, or
- AS2 can monitor its own availability and sent **monitoring results** to CS1

- A service used by a composite service needs to be replaced (e.g. stock checking service)
- A service is migrated to or starts using a different infrastructure (e.g. network provider, data management provider)
- The guaranteed terms of an SLA may change
- The monitoring components (e.g. event capturing or monitoring mechanisms) deployed by specific services may of different types and change

## Dynamic checks of SLA monitorability

- Is it possible to monitor the SLA given the monitoring capabilities of existing services?

## Dynamic decisions about the delegation of monitoring Tasks

- Identification of “best” source of events and monitor

## Dynamic identification and engagement of heterogeneous monitoring components

- Identify and engage dynamically monitors and event sensing components



## Dynamic updates of monitoring infrastructure

**Motivation & Required Capabilities**

**The SLA@SOI Monitoring Architecture**

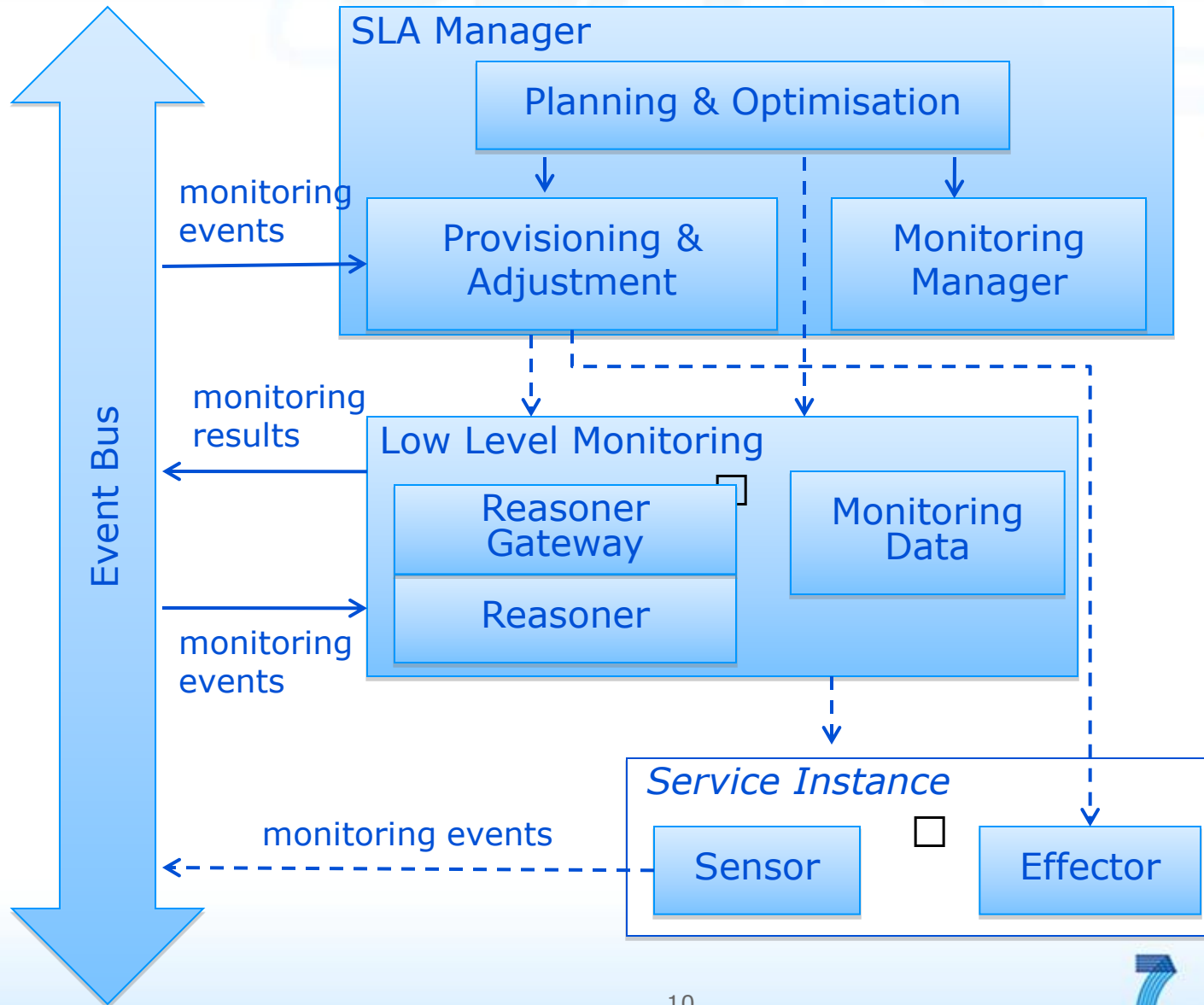
**Monitorability Check & Dynamic Set up**

**SLA Translation**

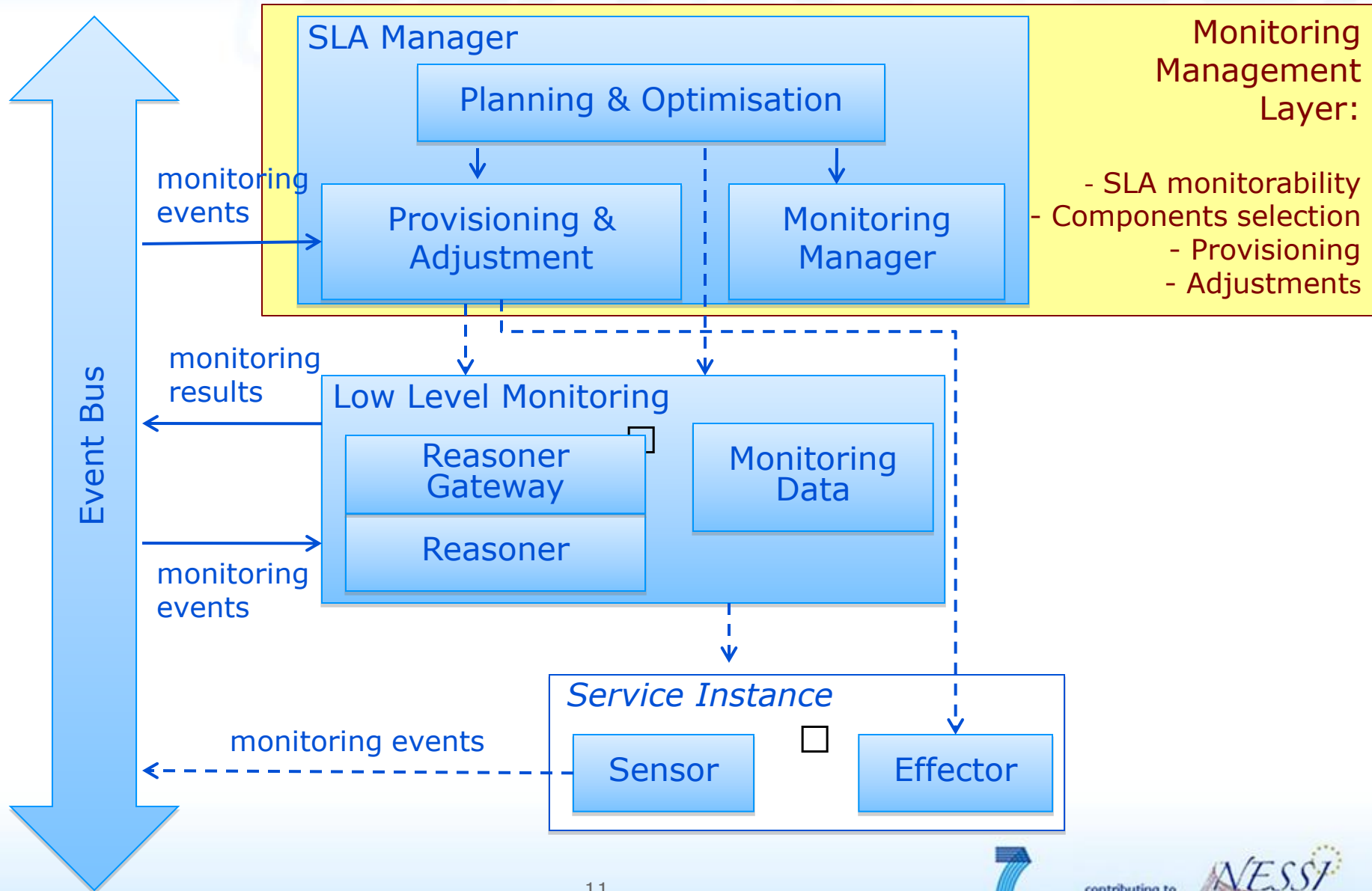
**Overview of Key Innovations**

**Open Issues & Ongoing Work**

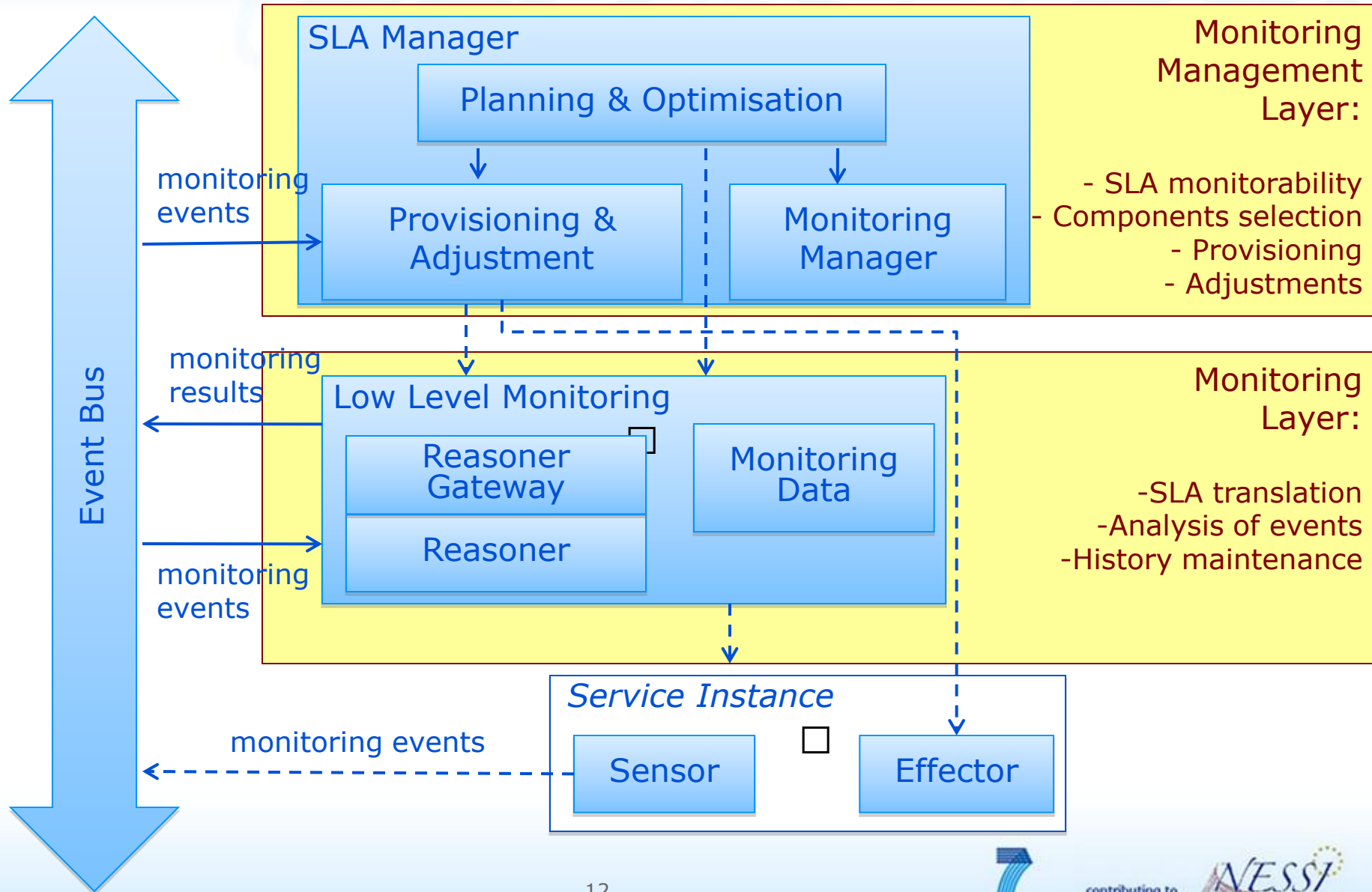
# The SLA@SOI monitoring architecture



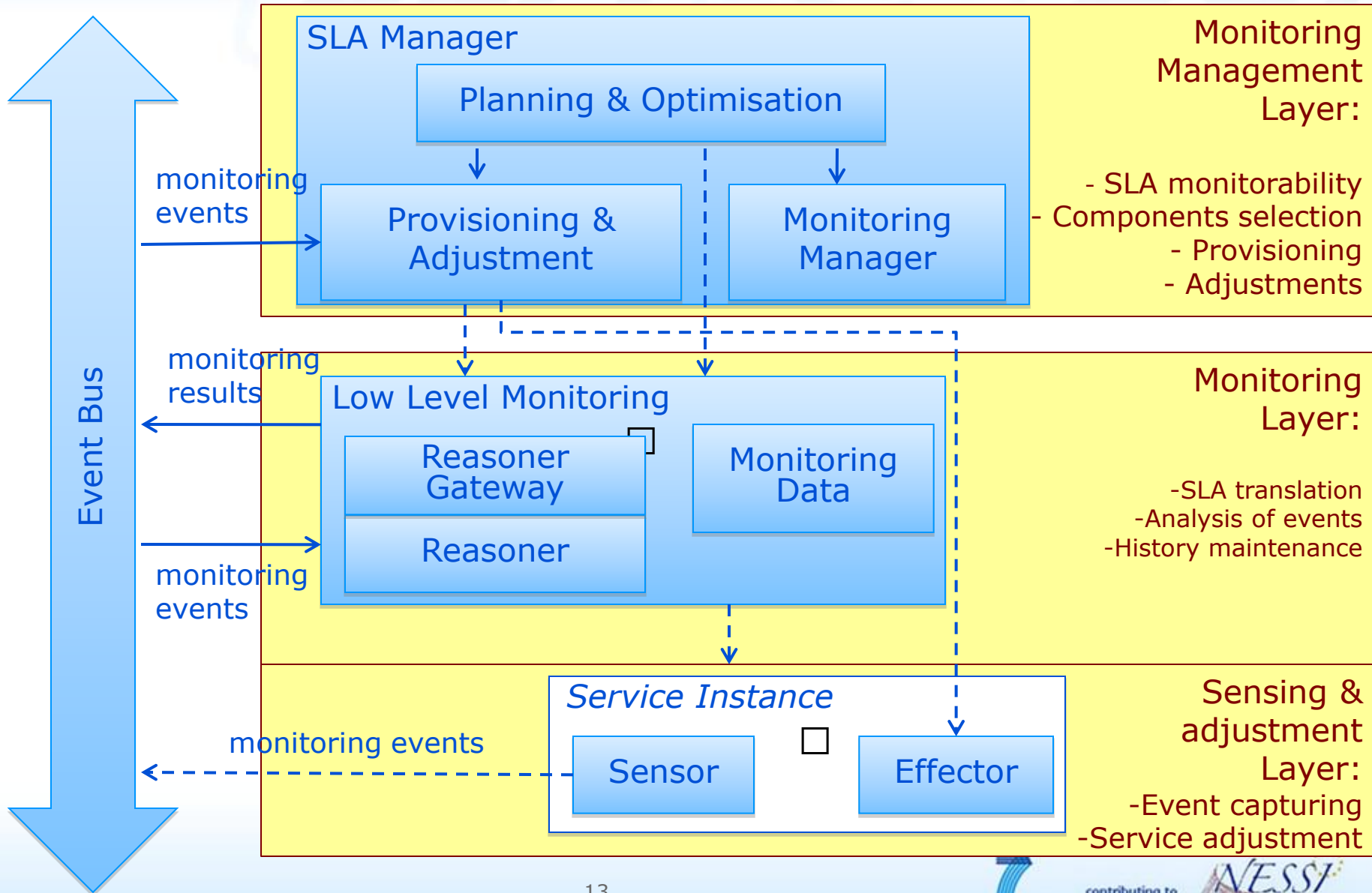
# The SLA@SOI monitoring architecture



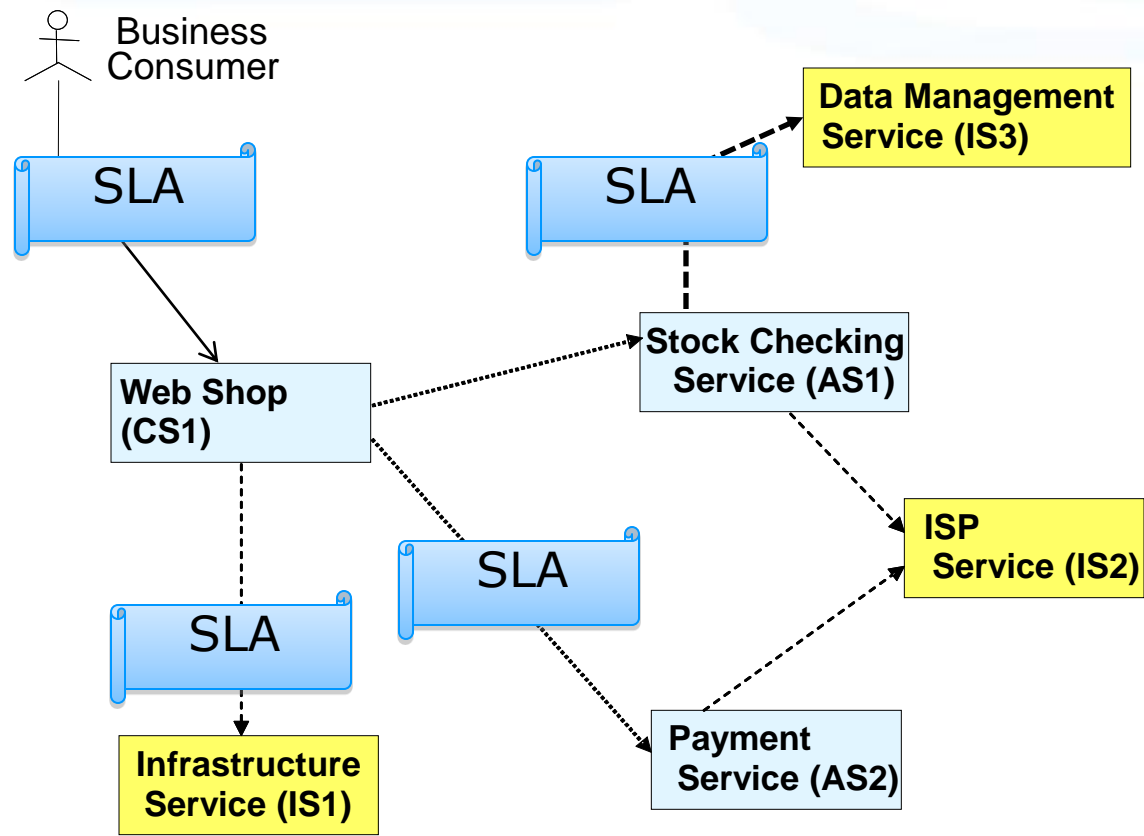
# The SLA@SOI monitoring architecture



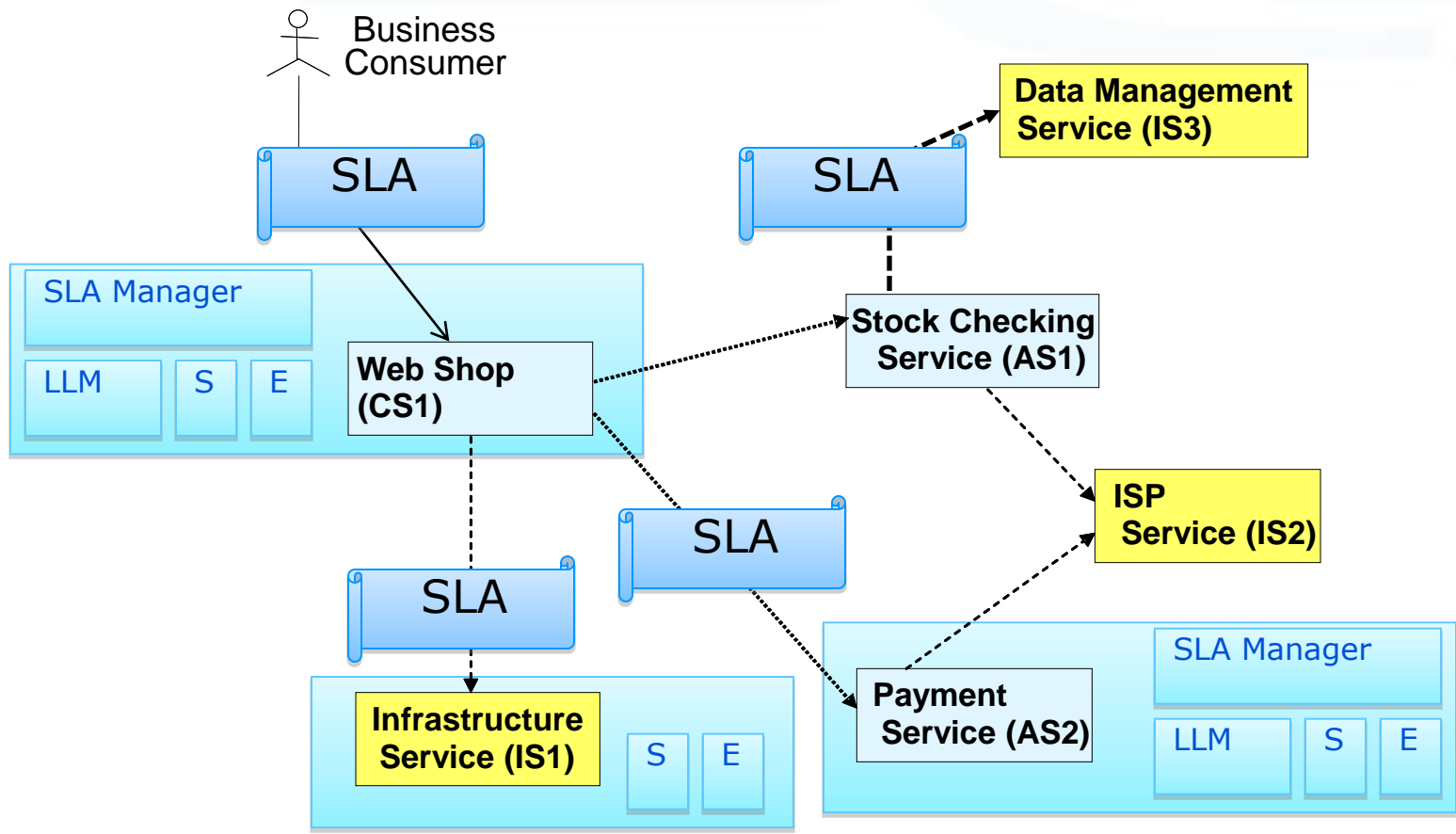
# The SLA@SOI monitoring architecture



# Example: Web Shop Service



# Example: Web shop Service



## Fixed elements

- SLA Manager
- Event structure & exchange
- Basic component interfaces & engagement protocols (e.g., those of Low level monitoring components, Sensors, Effectors)

## Variable elements

- Sensors
- Effectors
- Reasoners

## Monitoring Manager

- Checks the monitorability of SLAs
- Selects “optimal” monitoring configurations

## Low Level Monitors

- Translate high level SLAs to operational monitoring specifications acceptable by specific reasoners (aka monitors)
- Pass operational monitoring specifications to reasoners and receives data from them
- Maintain monitoring data

## Reasoners

- Execute SLA checks
- **Intrusive:** Instrumented into services; check properties at specific checkpoints (e.g., WSCoL monitor)
- **Non intrusive:** Run in parallel with the system checking if the events captured from it satisfy the SLA (e.g., EVEREST, Ganglia)

## Sensors

- Capture, encode and transmit information collected during the operation of a service instance in the form of events
- Domain specific
- Internal or external

## Effectors

- Modification of service instances and their configuration during service provisioning or runtime (e.g., stopping a service operation, changing a low level security policy for a service)
- Domain specific
- Internal or external

## Event Bus

- Publish/subscribe infrastructure managing event transmission
- Events: primitive monitoring information or results

**Motivation & Required Capabilities**

**The SLA@SOI Monitoring Architecture**

**Monitorability Check & Dynamic Set up**

**SLA Translation**

**Overview of Key Innovations**

**Open Issues & Ongoing Work**

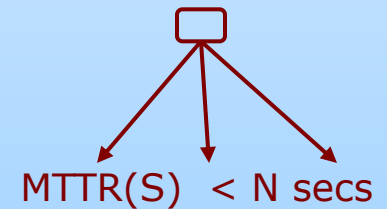
## Process

- Parse SLA terms & generate an abstract syntax tree (AST) of terms
- Match AST with monitoring features of available monitoring components (event sensors and monitors)
- Select “optimal” monitoring configuration

### SLA guarantee term:

$MTTR(S) < N \text{ secs}$

### Generate Term AST

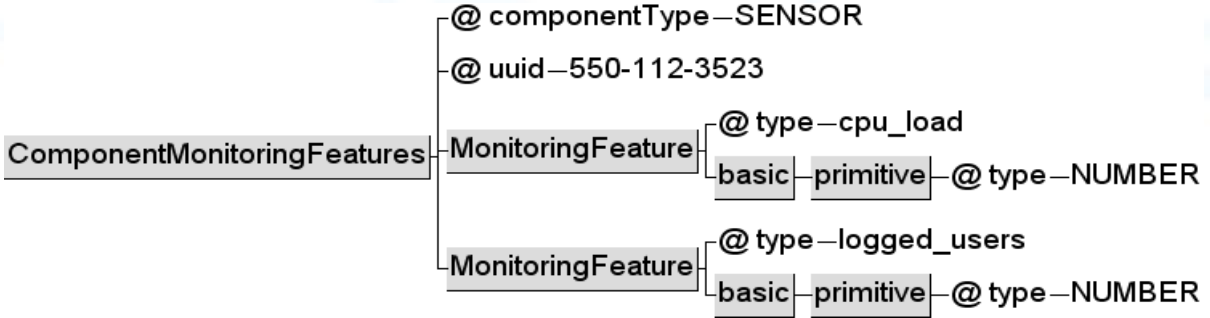


### Find reasoners able to:

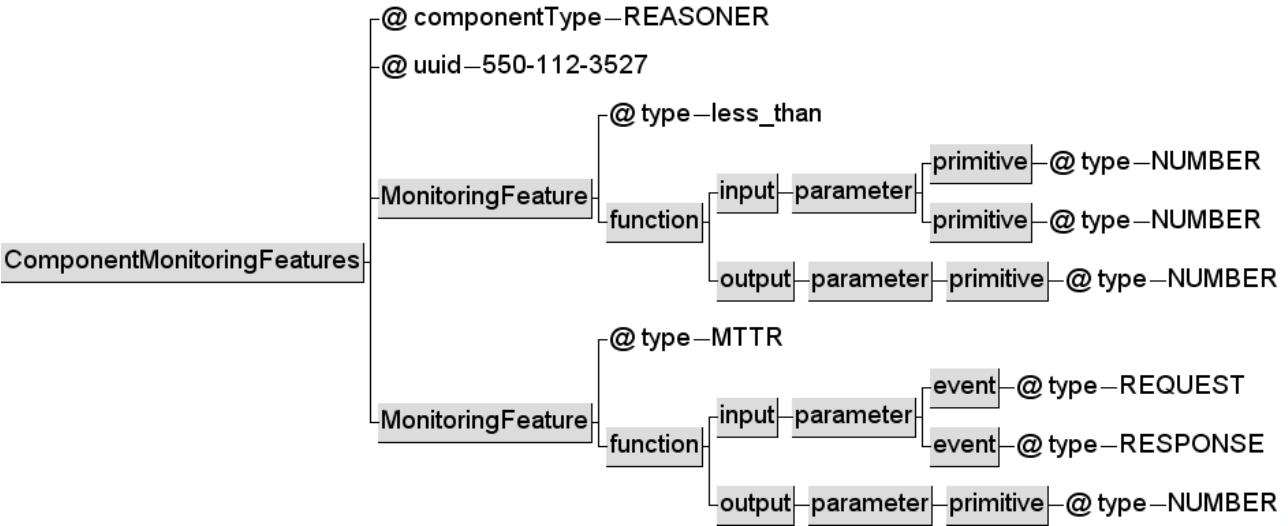
compute  $MTTR(S)$   
compute “<”

**& sensors able to**  
return primitive  
events for  $MTTR(S)$

## Sensor



## Reasoner



- Check term's monitorability and identify event sensors and monitors
- Select "optimal" monitoring components based on monitoring preferences
- Delegate responsibility to selected monitoring components
- Create event bus subscriptions to enable transmission of information (monitoring events & results)
- Initiate event sensors/reasoners

**Motivation & Required Capabilities**

**The SLA@SOI Monitoring Architecture**

**Monitorability Check & Dynamic Set up**

**SLA Translation**

**Overview of Key Innovations**

**Open Issues & Ongoing Work**

- **High level SLA syntax** → language of specific reasoner
- **EVEREST Reasoner Gateway:**  
Translation to *EC-Assertion* (based on *Event Calculus*):
  - ◇ **Monitoring rules** expressing the conditions to be monitored:  
 $B_{t_1} \Rightarrow H_{t_2}$  (if  $B_{t_1}$  is true then  $H_{t_2}$  must be true)
  - ◇ **Assumptions** keeping the values of state variables (fluents) required for the checks  
 $B_{t_1} \Rightarrow H_{t_2}$  (if  $B_{t_1}$  is true then deduce  $H_{t_2}$ )

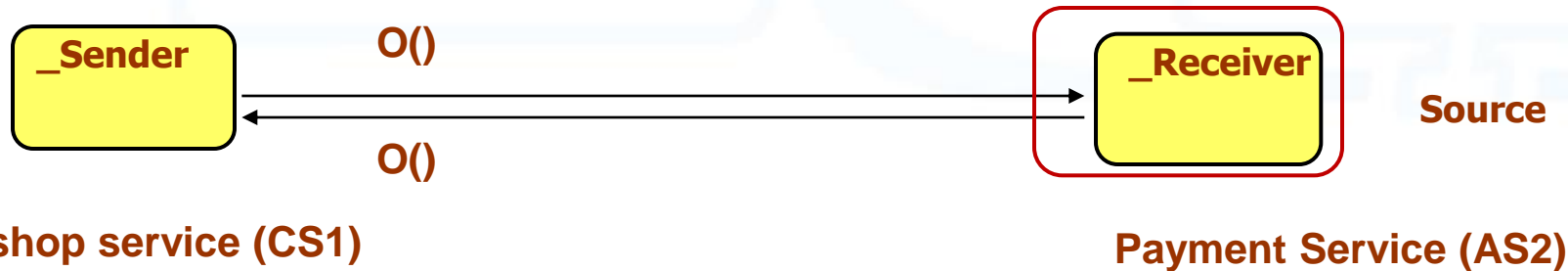
## Predefined predicates:

- **Happens( $e, t, \mathcal{R}(t_1, t_2)$ )** – occurrence of an instantaneous event  $e$  at some time  $t$  within the time range  $\mathcal{R}(t_1, t_2)$
- **Initiates( $e, f, t$ )** – fluent  $f$  starts to hold after the event  $e$  at time  $t$ .
- **Terminates( $e, f, t$ )** – fluent  $f$  ceases to hold after the event  $e$  occurs at time  $t$
- **HoldsAt( $f, t$ )** – fluent  $f$  holds at time  $t$ .

## Process overview

- **Parse** SLA(T) agreement terms and generate an abstract syntax tree (AST) for the term
- **Identify** *EC-Assertion* **formulae template** for maintaining the state variables required for the check of the SLA term
- **Instantiate** the template for the specific SLA term
- **Generate** monitoring rule for the SLA term

# SLA(T) translation to EC-Assertion



**SLA agreement term:**

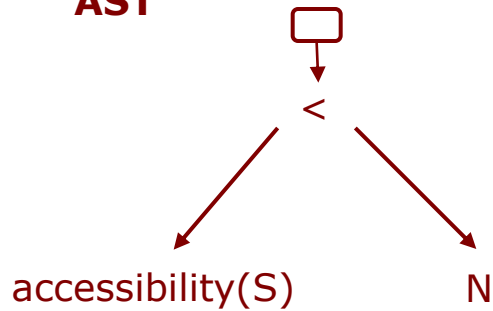
**Precondition:** NULL

**Guarantee state:**

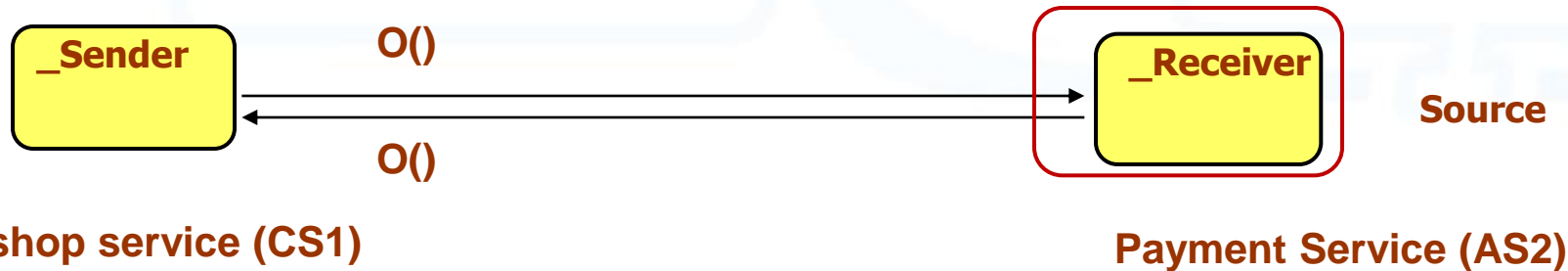
accessibility(S) < N ratio



**AST**



# SLA(T) translation to EC-Assertion



**SLA agreement term:**

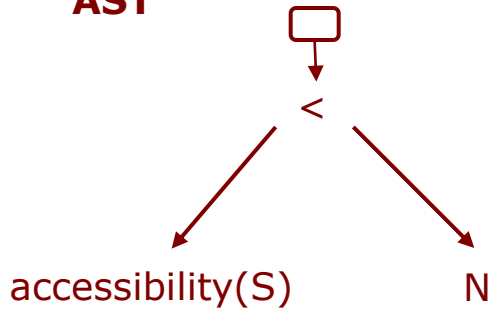
**Precondition:** NULL

**Guarantee state:**

accessibility(S) < N ratio



**AST**



## (1) Locate accessibility(S) template:

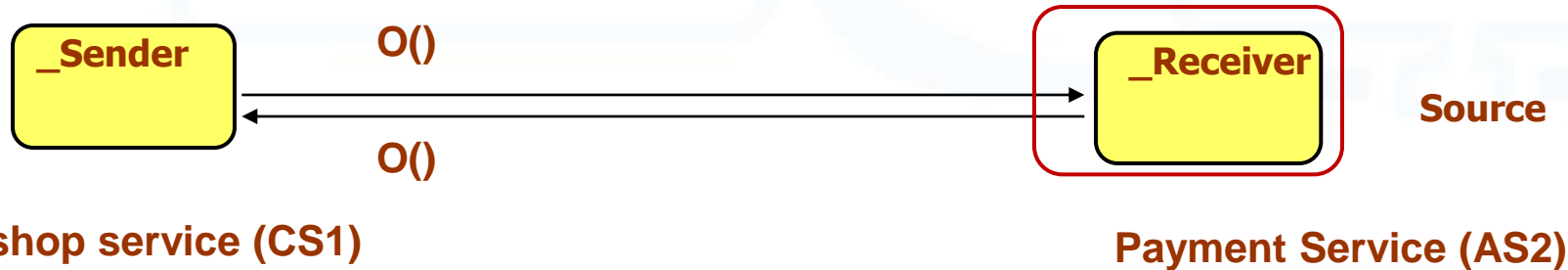
**A1:** Happens(e\_eID1, <S>, <R>, call(<O>), <R>), t1, R(t1,t1)) ^  
Happens(e\_eID2, <S>, <R>, res(<O>), <R>), t2, R(t1,t1+d)) ^  
**HoldsAt**(Accessibility(<S>, \_A, \_I),t2) =>  
**Initiates**(e\_eID2, ...), Accessibility(<S>, (\_A\*\_I)+1)/(\_I+1), \_I+1), t2+1)

**A2:** Happens(e\_eID1, <S>, <R>, call(<O>), <R>), t1, R(t1,t1)) ^  
**not Happens**(e\_eID2, <S>, <R>, res(<O>), <R>), t2, R(t1,t1+d)) ^  
**HoldsAt**(Accessibility(<S>, \_A, \_I),t2) =>  
**Initiates**(e\_eID1, ...), Accessibility(<S>, (\_A\*\_I)/(\_I+1), \_I+1), t2+1)

**Condition variable:** (Accessibility(<S>, \_A, \_I), \_A)

**Triggering condition:** default-period

# SLA(T) translation to EC-Assertion



## (2) Instantiate accessibility(S) template:

**SLA agreement term:**

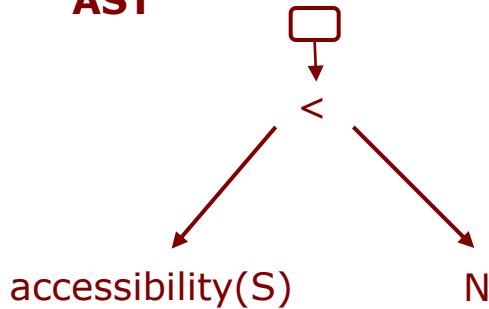
**Precondition:** NULL

**Guarantee state:**

accessibility(S) < N ratio



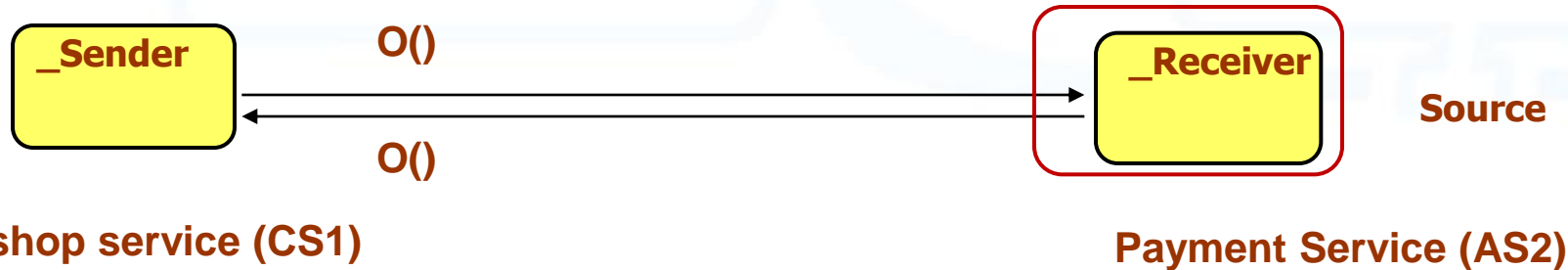
**AST**



**A1:** Happens(e\_eID1, CS1, AS2, call(\_O), AS2), t1, R(t1,t1))  $\wedge$   
 Happens(e\_eID2, CS1, AS2, res(\_O), AS2), t2, R(t1,t1+d))  $\wedge$   
 HoldsAt(Accessibility(AS2, \_A, \_I), t2)  $\Rightarrow$   
 Initiates(e\_eID2, ...), Accessibility(AS2, (\_A\*\_I)+1)/(\_I+1), \_I+1), t2+1)

**A2:** Happens(e\_eID1, CS1, AS2, call(\_O), AS2), t1, R(t1,t1))  $\wedge$   
 not Happens(e\_eID2, CS1, AS2, res(\_O), AS2), t2, R(t1,t1+d))  $\wedge$   
 HoldsAt(Accessibility(AS2, \_A, \_I), t2)  $\Rightarrow$   
 Initiates(e\_eID1, ...), Accessibility(AS2, (\_A\*\_I)/(\_I+1), \_I+1), t2+1)

# SLA(T) translation to EC-Assertion



## SLA agreement term:

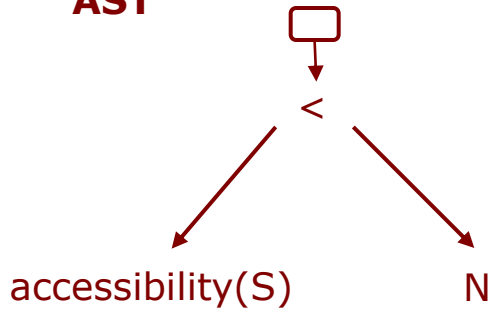
**Precondition:** NULL

**Guarantee state:**

accessibility(S) < N ratio



**AST**



## (2) Instantiate accessibility(S) template:

**A1:** Happens(e\_eID1, CS1, AS2, call(\_O), AS2), t1, R(t1,t1))  $\wedge$   
 Happens(e\_eID2, CS1, AS2, res(\_O), AS2), t2, R(t1,t1+d))  $\wedge$   
 HoldsAt(Accessibility(AS2, \_A, \_I), t2)  $\Rightarrow$   
 Initiates(e\_eID2, ..., Accessibility(AS2, (\_A\*\_I)+1)/(\_I+1), \_I+1), t2+1)

**A2:** Happens(e\_eID1, CS1, AS2, call(\_O), AS2), t1, R(t1,t1))  $\wedge$   
 not Happens(e\_eID2, CS1, AS2, res(\_O), AS2), t2, R(t1,t1+d))  $\wedge$   
 HoldsAt(Accessibility(AS2, \_A, \_I), t2)  $\Rightarrow$   
 Initiates(e\_eID1, ..., Accessibility(AS2, (\_A\*\_I)/(\_I+1), \_I+1), t2+1)

## (3) Create monitoring rule:

Condition variable: (Accessibility(<S>, \_A, \_I), \_A)

Triggering condition: default-period

**Rule:**

Happens(e\_eID1, Sys, Sys, default-period, Sys), t1, R(t1,t1) )  $\wedge$   
 HoldsAt(Accessibility(AS2, \_A, \_I), t1)  $\Rightarrow$  \_A <

**Motivation & Required Capabilities**

**The SLA@SOI Monitoring Architecture**

**Monitorability Check & Dynamic Set up**

**SLA Translation**

**Overview of Key Innovations**

**Open Issues & Ongoing Work**

# Key innovations of monitoring infrastructure



- Clear distinction between monitoring management and reasoning layer
- Support for heterogeneous monitors, sensors and effectors
- Support for dynamic adjustments of the architecture
- Static and dynamic checks of monitorability
- Configurable selection criteria for monitoring components
- Support for high level SLA language
- Formality (through mapping to Event Calculus)

**Motivation & Required Capabilities**

**The Monitoring Architecture**

**Monitorability Check & Dynamic Set up**

**SLA Translation**

**Overview of Key Innovations**

**Open Issues & Ongoing Work**

## Amendments of SLA(T)

For example:

- concept of "service failure"
- Timeouts

## Selection of "optimal" monitoring configurations

- Performance
- SLA Coverage
- Workload
- Trust

## Dynamic re-allocation of monitoring responsibilities

- Identification of transferrable SLA guarantee terms
- Transferring monitoring state across monitors

## Optimisation of translation process

e.g., Common assumptions for different SLA guarantee terms in  
EVEREST Reasoner Gateway

- Feedback for SLA(T) amendments through formal mapping to EC-Assertion
- Full support for SLA(T) grammar
- Elaboration of the monitoring configuration selection process

**Thank  
you!**