SLA Monitoring:
The SLA@SOI Monitoring Architecture

George Spanoudakis
CITY UNIVERSITY London

Date: 28/9/2010
Outline

Motivation & Required Capabilities
The SLA@SOI Monitoring Architecture
Monitorability Check & Dynamic Set up
SLA Translation
Overview of Key Innovations
Open Issues & Ongoing Work
Outline

Motivation & Required Capabilities

The SLA@SOI Monitoring Architecture
Monitorability Check & Dynamic Set up
SLA Translation
Overview of Key Innovations
Open Issues & Ongoing Work
Motivation

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

- Business Consumer
- Data Management Service (IS3)
- Stock Checking Service (AS1)
- ISP Service (IS2)
- Payment Service (AS2)
- Infrastructure Service (IS1)
- Web Shop (CS1)

deployed on/deploys
composed of
uses
Example: Web Shop Service

---

**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 msecs on average

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

---

- Web Shop (CS1) composed of
  - Stock Checking Service (AS1)
  - Payment Service (AS2)
  - ISP Service (IS2)
  - Data Management Service (IS3)

- Infrastructure Service (IS1) deployed on/deploys

---

- Business Consumer

---
Scenarios for adaptable SLA monitoring

• 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
Basic SLA monitoring Capabilities

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
Outline

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

- **SLA Manager**
  - Planning & Optimisation
  - Provisioning & Adjustment
  - Monitoring Manager

- **Low Level Monitoring**
  - Reasoner Gateway
  - Reasoner
  - Monitoring Data

- **Service Instance**
  - Sensor
  - Effector

Event Bus:
- Monitoring events
- Monitoring results
The SLA@SOI monitoring architecture

Monitoring Management Layer:
- SLA monitorability
- Components selection
- Provisioning
- Adjustments

Event Bus

SLA Manager

Planning & Optimisation

Provisioning & Adjustment

Monitoring Manager

Low Level Monitoring

Reasoner Gateway

Reasoner

Monitoring Data

Service Instance

Sensor

Effector

monitoring events

monitoring results

monitoring events

monitoring events
The SLA@SOI monitoring architecture

Monitoring Management Layer:
- SLA monitorability
- Components selection
- Provisioning
- Adjustments

Monitoring Layer:
- SLA translation
- Analysis of events
- History maintenance

Event Bus

monitoring events

monitoring results

monitoring events

monitoring events

SLA Manager
- Planning & Optimisation
- Provisioning & Adjustment
- Monitoring Manager

Low Level Monitoring
- Reasoner Gateway
- Reasoner
- Monitoring Data

Service Instance
- Sensor
- Effector
The SLA@SOI monitoring architecture

- SLA monitorability
- Components selection
- Provisioning
- Adjustments

Monitoring Management Layer:

- SLA translation
- Analysis of events
- History maintenance

Monitoring Layer:

- Event capturing
- Service adjustment

Sensing & adjustment Layer:

Monitoring

Low Level Monitoring

Reasoner Gateway

Reasoner

Monitoring Data

Service Instance

Sensor

Effector

SLA Manager

Planning & Optimisation

Provisioning & Adjustment

Monitoring Manager

Event Bus
Example: Web Shop Service

- Business Consumer
- SLA
  - Web Shop (CS1)
  - Infrastructure Service (IS1)
- SLA
  - Stock Checking Service (AS1)
  - Payment Service (AS2)
- SLA
  - ISP Service (IS2)
  - Data Management Service (IS3)
Example: Web shop Service

- Web Shop (CS1)
- Infrastructure Service (IS1)
- Stock Checking Service (AS1)
- Payment Service (AS2)
- ISP Service (IS2)
- Data Management Service (IS3)
- SLA Manager
- Business Consumer
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
Management & Monitoring Layer Components

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
Outline

Motivation & Required Capabilities
The SLA@SOI Monitoring Architecture
Monitorability Check & Dynamic Set up
SLA Translation
Overview of Key Innovations
Open Issues & Ongoing Work
Monitorability Test (Monitoring Manager)

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 guarantees term:

MTTR(S) < N secs

Generate Term AST

Find reasoners able to:
- compute MTTR(S)
- compute "<"
- & sensors able to return primitive events for MTTR(S)
Monitoring features

Sensor

Reasoner
Monitoring process initiation

- 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
Outline

Motivation & Required Capabilities
The SLA@SOI Monitoring Architecture
Monitorability Check & Dynamic Set up
SLA Translation
Overview of Key Innovations
Open Issues & Ongoing Work
SLA Translation (Reasoner Gateway)

• 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:
    $B_{t1} \Rightarrow H_{t2}$ (if $B_{t1}$ is true then $H_{t2}$ must be true)
  ◇ Assumptions keeping the values of state variables (fluents) required for the checks
    $B_{t1} \Rightarrow H_{t2}$ (if $B_{t1}$ is true then deduce $H_{t2}$)

Predefined predicates:
• Happens(e, t, $\mathcal{R}(t1,t2)$) – occurrence of an instantaneous event $e$ at some time $t$ within the time range $\mathcal{R}(t1,t2)$
• 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 $t$. 
SLA(T) translation to EC-Assertion

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 agreement term:
Precondition: NULL
Guarantee state:
accessibility(S) < N ratio

$\text{AST}$

$\text{accessibility}(S) < N$
SLA(T) translation to EC-Appearance

**SLA agreement term:**
- **Precondition:** NULL
- **Guarantee state:** accessibility(S) < N ratio

**AST**

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

**Triggering condition:** default-period

**Source**

![Diagram of Web shop service (CS1) and Payment Service (AS2)]

**Web shop service (CS1)**

**Payment Service (AS2)**

----

**(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)
SLA(T) translation to EC-Assertion

Web shop service (CS1)  
O()  
(Sender)

O()  
(Receiver)

Payment Service (AS2)

Source

(2) Instantiate accessibility(S) template:

A1: Happens(e(_eID1, CS1, AS2, call(_O), AS2), t1, R(t1,t1)) \(\land\)

Happens(e(_eID2, CS1, AS2, res(_O), AS2), t2, R(t1,t1+d)) \(\land\)

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)) \(\land\)

not Happens(e(_eID2, CS1, AS2, res(_O), AS2), t2, R(t1,t1+d)) \(\land\)

HoldsAt(Accessibility(AS2, _A, _I),t2)) \(\Rightarrow\)

Initiates(e(_eID1, ...), Accessibility(AS2, (_A* _I)/(_I+1), _I+1), t2+1)

SLA agreement term: 
Precondition: NULL
Guarantee state: 
accessibility(S) < N ratio

AST

accessibility(S)  N
SLA(T) translation to EC-Automation

**Web shop service (CS1)**

**Payment Service (AS2)**

**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)) \land
Happens(e(_eID2, CS1, AS2, res(_O), AS2), t2, R(t1,t1+d)) \land
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)) \land
not Happens(e(_eID2, CS1, AS2, res(_O), AS2), t2, R(t1,t1+d)) \land
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:***
\[ \text{Happens}(e(_eID1, Sys, Sys, default-period, Sys),t1, R(t1,t1)) \land
\text{HoldsAt}(Accessibility(AS2, _A, _I),t1)) \Rightarrow _A < \]
Outline

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)
Outline

Motivation & Required Capabilities
The Monitoring Architecture
Monitorability Check & Dynamic Set up
SLA Translation
Overview of Key Innovations
Open Issues & Ongoing Work
Open Issues

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
Ongoing work

- 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!