Towards Autonomic Enterprise Service Bus

MAASC’11, 12 May 2011 Paris
Denis Morand (Grenoble Univ./Schneider-Electric)
Issac-Noé García Garza (Grenoble University)
Philippe Lalanda (Grenoble University)
Software Oriented Computing (SOC)

- P. Papazoglou: *Service-Oriented Computing (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications/solutions.*

- Service is a pair of [Service interface - Service implement.]
  - Service interface (platform independent) focused on
    - Functional information,
    - Location,
    - Invocation
  - Service implementation
    - black box
Architecture

1. Publication

2. Discovery

3. Binding & invocation

• Heterogeneity is hidden
• Loosely coupled
• Late binding
• Substituability

Service consumer

Broker

Service provider

Service description

Service Description
Plan

• Introduction
• Mediation
• Cilia
• Towards Autonomic Cilia
• Conclusion
Mediation

• Service Mediation adds operations
  – Communication : Protocol transformation, content base routing
  – Services : Coordination, Service exchange (syntactic, semantic), Non functional services (QoS, logging, monitoring, security, persistency...)

  – Benefits
    • Facilitates evolution of application, non functional services may added transparently
    • Improves scalability of the system.
Enterprise Service Bus

- An ESB provides runtime environment mediating Web Service applications
Plan

- Introduction
- Mediation
- Cilia
- Towards Autonomic Cilia
- Conclusion
Cilia overall approach

Specification

Execution

represents

Mediation chain

Design model (component based)

Execution model (iPOJO based)

Client

Scheduler

Processor

Dispatcher

Service

Service

Service

Cilia mediation

Denis Morand  28/04/11
Plan

• Introduction
• Mediation
• Cilia
• Towards Autonomic Cilia
• Conclusion
Towards Cilia autonomic

- To make Cilia autonomic we identified tasks to be done
  1. Information to be monitored
  2. Runtime Capture of information
  3. Define a model to store these aspects
Plan

• Introduction
• Mediation
• Cilia
• Towards Autonomic Cilia
  – Autonomic computing goals
  – Information to Monitor
  – Runtime Capture information
  – Meta-model
• Conclusion
Goals of autonomic computing

• The goals autonomic computing is to reduce the cost of service following axes :

1. Self-Management
2. Efficient use of «available» resources and capacities.
3. Hide complexity
Information to monitor

• We actually distinguish three kinds of information to monitor
Level Framework

CPU load
Java heap

#Threads in use

#Buffer_In in use
#Buffer_Out in use

Client

Mediation chain

Execution platform

Service

Service

Service
Level Chain

Propagation delay

Data Sources

Cilia mediation Application

Data Consumers

#incoming messages

Statistics (mean, max, mi, n, stddev)

#outgoing messages
Level Mediator

Processing delay

Transmission delay

Data Sources

JDBC

SOAP

HTTP

Cilia mediation Application

Data Consumers

JMS

JDBC

#msg scheduler (ok, error)
#msg processor (ok, error)
#msg dispatcher (ok, error)

Statistics
(mean, max, mi, n, stdev)
Plan

• Introduction
• Mediation
• Cilia
• Towards Autonomic Cilia
  – Autonomic computing goals
  – Information to Monitor
  – Runtime Capture information
  – Meta-model
• Conclusion
How to Monitor an ESB?

- **State Variable principle (inspired from work in control theory)**

![Diagram showing state variables and system monitoring]

- System monitored through 1 state variable
- System monitored through 2 state variables
How to react on the system?

• Control Variable principle (inspired from work in control theory)
Plan

• Introduction
• Mediation
• Cilia

• Towards Autonomic Cilia
  – Autonomic computing goals
  – Information to Monitor
  – Runtime Capture information
  – Meta-model

• Conclusion
Meta-model
State variable & Action variable

Condition to store the measure (LDAP filter)

Viability zone

Measurement
+DoubleValue : double
+StringValue : string
+timeStamp : long

Viability
+getQoS() : Metrics

Knowledge
+name : string

CorrectiveAction
-Timeout : long
+getAction()
+setAction()

Observation

Condition
+setFilter() : string
+getFilter() : string

0..* matches

Viability

Store

Measure Timestamped

Value to set
Observation

- An Observation is composed by a list of Measurements
  - Observation is the container of the state variable values.
  - A Measurement is stored if and only if the condition matches.
  - Measurement are stored in circular fifo manner.
Condition

- A Condition is a filter specified by a LDAP filter criteria

- We distinguish two kinds of filtering
  - Filtering by value
  - Filtering by time

- The filter can be a mixed of filtering by value and filtering by time.
Filtering by Time

- The filter can use any condition of the following attributes
- \textit{TIME\_ELAPSED} : Time between the previous Measurement stored and the actual.
- \textit{TIME\_CURRENT} : Timestamp of the current measurement
- \textit{TIME\_PREVIOUS} : Timestamp of the previous measurement.
  - Example : A measurement is stored with a maximum 60 per second (unit is ms, once per second)
    \[(TIME\_ELAPSED \geq 1000)\]
  Changing the test, allows increasing or reducing the sampling rate.
Filtering by Value

- The filter can use any condition of the following attributes
- `VALUE_CURRENT` : Current value of the data
- `VALUE_PREVIOUS` : Previous value of the data
- `DELTA_ABSOLUTE` : Positive difference between current value and previous value
  - Example : A measurement is stored only if the new value is different from the previous.
    \[
    (\text{DELTA\_ABSOLUTE} > 0)
    \]
Viability

- A state variable comes with a validity interval
- Exemple:
  \[10 \leq \text{StateVar.data} \leq 15\] -> OK

\[5 \leq \text{StateVar.data} < 10\] -> Low “Warning”
\[x \leq \text{StateVar.data} < 5\] -> Very Low “danger”
Plan

• Introduction
• Mediation
• Cilia
• Towards Autonomic Cilia
• Conclusion
Conclusion

• The state variables and control variables inspired from Control theory
  – Allows to follow low level programming aspects
  – Ability to adjust threshold values and filter “on demand”
    • Flow control management and viability (stability)

• Next
  – Design and implementation of higher-level autonomic decisions
    • Topology modification, change/replace on fly a mediator ...
Thank you!