A Sensor-based Approach to Symptom Recognition for Autonomic Systems

Prepared by Faisal Sibai for CS 895

Presentation Agenda

- Introduction and proposal
- Background and related work
- System Architecture
- Scenario
- Conclusion
Introduction

• Today’s systems run in a complex way that leads to management complexities.

• To facilitate task management, problem determination, performance tuning and others, Autonomic Computing comes into the picture.

• In this paper we concentrate on self-healing and partially self-management. (solving problems when something wrong is detected)

• Events – important for this paper:
  o Single event (database failed access attempt)
  o Complex event are more than one event that are indicative of problems within the system that could lead to potential failure

• A symptom is a form of knowledge that indicates a problem. In AC symptoms are recognized by the monitoring system and then are compared against database of known symptom patterns and successful resolutions to automatically find a solution.
• Monitoring subsystem based on an event-driven sensor architecture.

• A lightweight, flexible, standards-base approach to symptom recognition in an autonomic environment. (discussion is geared towards web service but is virtually applicable to any distributed environment)

• Major contributions:
  o The specification of a highly distributed, policy-based sensor framework for management
  o The proposal of a novel use of complex event processing
  o The adoption of the OASIS Web Services Distributed Management (WSDM) standard to illustrate the relevance and applicability of this standard to systems management.

• Approach is based on WSDM using WS-Notification and WEF Messages (will be discussed in the following slide)
• **Normal Web service**
  o **Definition**: "a software system designed to support interoperable machine to machine interaction over a network".
  o Service providers and consumers communicate using XML messages that follow the SOAP standard
  o **Maximize** Interoperability and minimize degree of coupling

• **WS-Notification framework (WSN):**
  o Based on event notification
  o A mechanism for a service to distribute information to other services, without prior knowledge of the receivers
  o **WS-Base-Notification** defines the interface that WS-Notification consumers and producers should expose
  o **WS-Topic** addresses the definition and use of topics to which Web Services consumers can subscribe

• **Web Service Distributed Management (WSDM):**
  o A standard for the management of Web-based services
  o Allows a manager service to communicate with any other service that is WSDM-compliant
  o **MUWS** defines the interfaces to represent and access the manageable functions of available resources. Also provides a standard management event format (WEF) to improve interoperability and correlation between events
  o **MOWS** standard defines how to manage Web Services resources as well as how to describe and access the manageable functions using MUWS
Related work

- Monitoring and symptom recognition are key steps in self-healing
  - **Nehme** A Framework for self-healing DBMSs, integrated inside the DBMS, actively detects problem symptoms, matches against problems’ signatures and performs corrective actions
  - **Pernici and Rosati** Pattern recognition to automatically learn repair strategies to support self-healing of Web services
  - **WS-Diamond** Framework supporting the execution of self-healing Web Services

- Autonomic fault detection and self-healing techniques for networks
  - **AHSEN** (Autonomic Healing-Based Self Management Engine for Network Management) is a policy-based management system that monitors and heals a network system using context-driven self-management functions

- Paper proposes a hierarchical approach to symptom recognition where information is passed from lower level sensors and is correlated and analyzed by higher level sensors. **Lapouchnian** is similar but the paper’s approach is policy-based and does not require any changes to the monitored system

- **Assumption**: A standard management interface for managed resources (using WSDM) and the monitoring system is layered on top of these components.
System Architecture

- Hierarchy of sensors
- Communication via event notification
- Sensors take events from other sensors, process and emit new event notifications (complex event)
- These events are taken by higher sensors
- Notifications contain symptoms
- Sensor controlled by its policy
- Sensors are identical except for their policy
- Individual sensors are implemented as WSDM (exposing a standard management interfaces, publish/subscriber)

Figure 1. Sensor architecture
P/S allows a sensor to subscribe to topics produced by other sensors or WSDM entities and receive notifications of events published to the topics.

Topics is a way to organize and categorize items of interest for subscription (example: response time).

Sensors provide multiple topics to which they may publish and they may subscribe to multiple topics.

Behavior of sensor is defined by its policy.

Policy defines topics produced and subscribed by sensor as well as complex events.
A complex events contains a description of a symptom recognized by the sensor.

Policy defines conditions under which complex event should be generated (Condition/action pattern).

Example: “the response time has increased more than 10 percent over the past 5 minutes”

Event notifications are collected by the sensor’s event collector in original XML format stored in ER.

This triggers the event analyzer that evaluates new events based on the C/A pattern (using Xquery).

Query returns list of matches, null is returned if not satisfied.
If there are matches a complex event notification is generated by the complex events generator and then published.

In our previous example of response time, if the XQuery returns a non-empty set then a complex event is generated and published.

The monitoring system consists of a hierarchy of simple, cooperating sensors, each with a very narrow view of the system.

Lowest level of the hierarchy, sensors consume event notifications either directly from a managed resource or from another sensor hierarchy.

Become meaningful (flow-up)
**Scenario**

- Prototype of symptom recognition system was implemented using Eclipse TPTP, Apache Muse and IBM DB2 and it has been integrated into a previously implemented system called Autonomic Web Services Environment (AWSE).
- AWSE is a framework that was developed for testing and showcasing ideas for autonomic management of computing systems.
- Their system monitors the web server, the web services and the DBMS server.

![AWSE Scenario Diagram](image-url)
Scenario

- Condition action statement specified by the following Xquery for response time example:

```xquery
declare namespace muws1="http://docs.oasis-open.org/wsdm/muws1-2.xsd";
declare namespace muws2="http://docs.oasis-open.org/wsdm/muws2-2.xsd";
for $a in db2-fn:xmlcolumn('RESPTIME.EVENT')/muws1:ManagementEvent, $b in db2-fn:xmlcolumn('RESPTIME.EVENT')/muws1:ManagementEvent
where xs:dateTime(fn:string($b/@ReportTime)) >
xs:dateTime(fn:string($a/@ReportTime)) and
xs:dateTime(fn:string($b/@ReportTime)) &lt;
xs:dateTime(fn:string($a/@ReportTime)) +
xdt:dayTimeDuration("PT5M") and
fn:number($b/muws2:Situation/muws2:Message)-
fn:number($a/muws2:Situation/muws2:Message) &gt; 5
return &lt;x&gt;${$a/muws1:EventId/text()},
{$b/muws1:EventId/text()}&lt;/x&gt;
```

- The top level management entity receives a notification from a sensor containing a list of current system symptoms
Conclusions and future work

- Systems are becoming increasingly complex and difficult to manage

- Web services and the WSDM standards provide better ways to monitor and manage web service enabled systems.

- They have monitored components using a sensor-based approach

- Resources were monitored and suggestions were provided using their approach

- Things to explore:
  - Scalability and overhead of sensor communication in large numbers of sensors
  - How will sensor failure be handled and detected