An Assessment of a Model for Error Processing in the CMS Data Acquisition System

Schahram Dustdar[1], Johannes Gutleber[2], Roland Moser[1,2], Luciano Orsini[2]
[1]Technical University of Vienna, Vienna, Austria
[2]CERN, Geneva, Switzerland

ABSTRACT

The CMS Data Acquisition System consists of O(20000) of interdependent services. A monitoring system providing exception and application-specific data is essential for the operation of this cluster.
Due to the number of involved services the amount of monitoring data is higher than a human operator can handle efficiently. Thus moving the expert-knowledge for error analysis from the operator to a dedicated system is a natural choice. This reduces the number of notifications to the operator for simpler visualization and provides meaningful error cause descriptions and suggestions for possible countermeasures.
This paper discusses an architecture of a workflow-based hierarchical error analysis system based on Guardians for the CMS Data Acquisition System. Guardians provide a common interface for error analysis of a specific service or subsystem. To provide effective and complete error analysis, the requirements regarding information sources, monitoring and configuration, are analyzed. Formats for common notification types are defined and a generic Guardian based on Event-Condition-Action rules is presented as a proof-of-concept.

INTRODUCTION

The CMS Data Acquisition System consists of O(20000) interdependent services. A monitoring system providing exception and application-specific data is essential for the operation of this cluster.
Due to the number of involved services the amount of monitoring data is higher than a human operator can handle efficiently. Thus moving expert-knowledge for error analysis from the operator to a dedicated **Error Processing system** is a natural choice. This reduces the number of notifications to the operator for simpler visualization and provides meaningful error cause descriptions and suggestions for possible countermeasures.

TECHNOLOGY

The CMS data acquisition system follows a service oriented architecture (SOA) [1] where each service provides a SOAP control interface. A fundamental subsystem for error processing is a scalable monitoring system. The XDAQ monitoring and alarming system (XMAS) [2] infrastructure based on a scalable and distributed publish/subscribe eventing system [3] handles currently O(100000) notifications per second.
Continuing with a service based approach we implemented an **error processing system with Web Workflows**. We chose to use the **ActiveBPEL** [4] workflow engine which combines workflows with SOAP based web services and can be extended to support additional requirements on the system at hand.

Information Sources

Sources of Information can be either run-time information (actual condition) or configuration information (nominal condition). Run-time information can be (Figure 3):

- **State information** contains information about the actual state of services. With hierarchical states as defined in ASAP [5] we can impose general states for visualization and error processing and allow refinement when necessary for control (Figure 4).
- **Error information** covers exceptions which could not be handled locally by the service. It embeds a complete exception trace for debugging.
- **Service information** contains dynamic data ranging from statistics to configuration data not known ad-hoc. It is freely definable.

Configuration information can be **hardware Information** that describes the setup of hosts, devices and networks or **software information** describing applications, services and communication endpoints.

ERROR PROCESSING ARCHITECTURE

A high-level error processing system is responsible to detect the cause of errors in startup and operation of the monitored system. Therefore it analyzes differences between actual and nominal condition.
The **Error Processor** is an intermediate which subscribes to the monitoring for retrieving error notifications and forwards them to intermediate error analysis components, called **Guardians** (Figure 1). Based on their responses it forwards notification to the operator and monitoring system accordingly. If exceptions could not be matched to an error cause it informs the operator to refine the rules for notifications based on their unique identifier.
Guardians are hierarchically ordered error analysis components and contain expert knowledge about specific services or subsystems (Figure 2). Their SOAP interface includes a list of error notifications and a list of URLs of monitoring data servers which may be queried for more information. They respond with notifications for an operator if an error cause could be identified or an error notification which is propagated to a higher level Guardian. The response additionally encloses a list of matched notification identifiers.
We implemented error processing using **BPEL** as it already provides powerful languages for filtering and querying XML data (XPath and XQuery). We use these features to implement **event/condition/action (ECA) rules** (Figure 5).

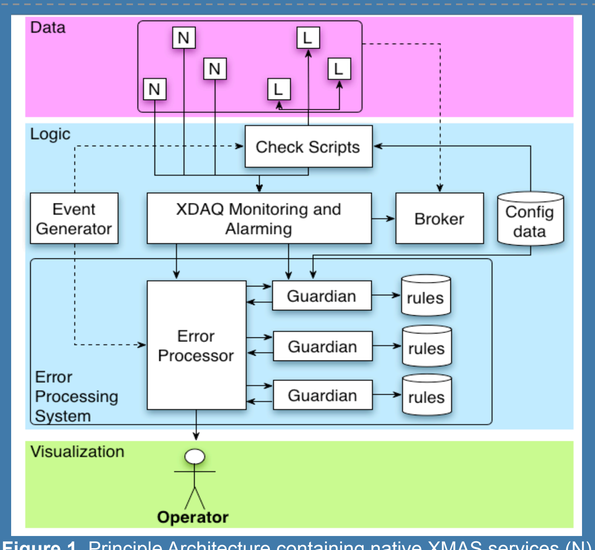

Figure 1. Principle Architecture containing native XMAS services (N) and legacy services (L).

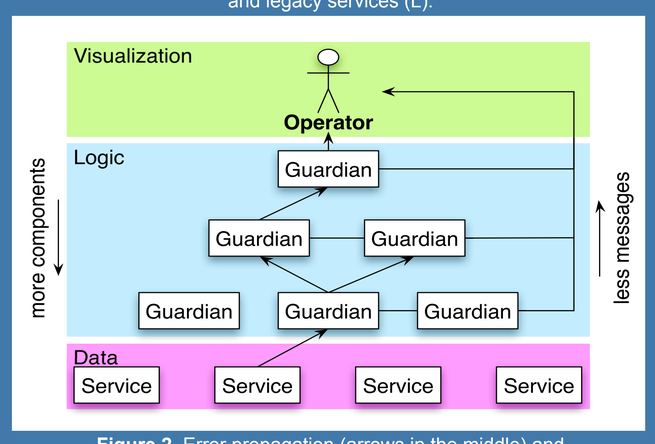

Figure 2. Error propagation (arrows in the middle) and operator notifications (arrows on the right).

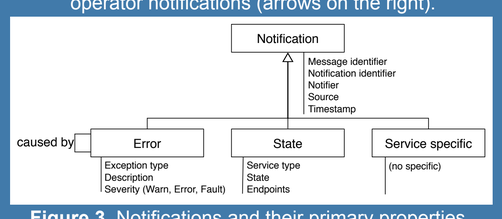

Figure 3. Notifications and their primary properties.

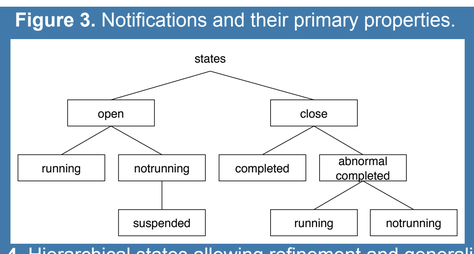

Figure 4. Hierarchical states allowing refinement and generalization.

DRAWBACKS

During evaluation of existing workflow engines we identified some **shortcomings of BPEL** and missing components necessary for integration with our system:

- BPEL workflows can only be triggered through SOAP messages and not through timers or even more complex rules.
- ActiveBPEL natively supports only SOAP based protocols.
- BPEL does not support to model an organizational perspective and mapping of services to invoke activities must be modelled explicitely.

ENHANCEMENTS

To overcome those shortcomings we implemented several additional services:
Event Generator is a service which sends SOAP messages based on predefined rules. Rules may match on workflow engine, timing and external user events.
Broker is a component for dynamically allocating resources and services upon QoS requests. It works with models for different use cases. A model for the monitoring system implements a load balancer for periodically allocating monitoring services to O(20000) services reporting monitoring information based on CPU load and thus helps achieving a scalable monitoring infrastructure.
Integration: As not all services publish directly into XMAS we need custom workflow scripts querying their states over SSH and publishing their information into XMAS through SOAP messages. In addition some services use a custom, binary protocol for performance reasons.
Although WSDL allows to define interfaces independent of transport protocols, we needed to integrate the workflow engine with custom protocols and execution of commands over SSH. ActiveBPEL provides an InvokationHandler which translates between internal workflow engine data representation (XML) and custom formats and therefore allow seamless integration with our system at hand.

REFERENCES

[1] “Web Services Architecture” http://www.w3.org/TR/ws-arch/
[2] “Monitoring the CMS Data Acquisition System”, CHEP 2009.
[3] “Web Services Eventing (WS-Eventing)” http://www.w3.org/Submission/WS-Eventing/
[4] ActiveBPEL Engine – official homepage http://www.activevos.com/community-open-source.php
[5] “Asynchonous Service Access Protocol (ASAP) Version 1.0” http://www.oasis-open.org/committees/asap/

```
<eca xmlns:tns="http://xdaq.web.cern.ch/xdaq/wsdl/2008/guardianeca-10.wsdl">
 <source type='flashlist' name='diskInfo'>urn:xdaq-flashlist:diskInfo</source>
 <rule>
  <condition>
/*/source/diskInfo/table/rows[ diskUsage/rows[xs:double(usePercent/text())>90] ]
  </condition>
  <action>
   <inform>
    <message>free disk space below 10 percent</message>
    <services source="condition">/*/rows/context</services>
   </inform>
  </action>
 </rule>
</eca>
```

Figure 5. ECA rules for generic Guardian detecting low disk space.

CONTACTS

Name: Roland Moser
Organization: CERN
Email: Roland.Moser@cern.ch
Phone: +41 2276 70808
http://xdaq.web.cern.ch

Name: Johannes Gutleber
Organization: CERN
Email: gutleber@cern.ch
Phone: +41 2276 71536
http://xdaq.web.cern.ch

Name: Luciano Orsini
Organization: CERN
Email: Luciano.Orsini@cern.ch
Phone: +41 2276 71615
http://xdaq.web.cern.ch

Name: Schahram Dustdar
Organization: TU Wien
Email: dustdar@infosys.tuwien.ac.at
Phone: +43 1 58801-18414
http://www.infosys.tuwien.ac.at/staff/sd