Distributed error and alarm processing in the CMS data acquisition system

G. Bauer [7], U. Behrens [1], M. Bowen [2], J. Branson [5], S. Bukowiec [2], S. Cittolin [3,5], J. A. Coarasa [2], C. Deldicque [2], M. Dobson [2] , A. Dupont [2], S. Erhan [4], A. Flossdorf [1], D. Gigi [2], F. Glege [2], R. Gomez-Reino [2], C. Hartl [3], D. Hatton [1], J. Hegeman [2], A. Holzner [5], Y. L. Hwong [2], L. Masetti [2], F. Meijers [2], E. Meschi [2], R. K. Mommsen [6], V. O'Dell [6], L.Orsini [2], C. Paus [7], A. Petrucci [2], M. Pieri [5], G. Polese [2], A. Racz [2], O. Raginel [7], H. Sakulin [2], M. Sani [5], C. Schwick [2], D. Shpakov [6], S. Simon [2], A. Spataru [2], K. Sumorok [7]

[1] DESY, Hamburg, Germany; [2] CERN, Geneva, Switzerland; [3] Eidgenössische Technische Hochschule , Zurich, Switzerland; [4] University of California, Los Angeles , Los Angeles, California, USA [5] University of California, San Diego, San Diego, California, USA; [6] FNAL, Chicago, Illinois, USA; [7] Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

ABSTRACT

The Error and Alarm system for the data acquisition of the Compact Muon Solenoid (CMS) at CERN is successfully used for the physics runs at Large Hadron Collider (LHC) during the first three years of activities. Error and alarm processing entails the notification, collection, storage and visualization of all exceptional conditions occurring in the highly distributed CMS online system using a uniform scheme. Alerts and reports are shown on-line by web application facilities that map them to graphical models of the system as defined by the user. A persistency service keeps history of all exceptions occurred, allowing subsequent retrieval of user defined time windows of events for later playback or analysis. The paper describes the architecture and the technologies used and deals with operational aspects during the first years of LHC. In particular it focuses on performance, stability and integration with the CMS sub-detectors.

REFERENCES

1. The CMS Collaboration, The Compact Muon Solenoid Technical Proposal, CERN/ LHCC94-38 (1994)
2. The LHC Study Group, The Large Hadron Collider Conceptual Design Report, CERN/ AC95-05 (1995).
3. The CMS Collaboration, The Trigger and Data Aquisition project, CERN/LHCC 2002-26, 15 December 2002.
4. Antchev G et al 2001, The Data Acquisition System for the CMS Experiment at LHC *in Proc. 7th Intl. Conf. Adv. Tech. and Particle Phys. Villa Olmo, Como, Italy (Oct. 15-19, 2001)* World Scientific Publishers (ISBN 981-238-180-5)
5. Bauer G et al 2008 *CMS DAQ Event Builder Based on Gigabit Ethernet* IEEE Trans. Nucl. Sci. **55(1)** 198-202
6. Gutleber J, Murray S, Orsini L 2003 Towards a homogeneous architecture for high-energy physics data acquisition systems *Elsevier Comp. Phys. Comm.* **153(2)** 155-163
7. Parnas D L 1979 Designing Software for Ease of Extension and Contraction, *IEEE Trans. Softw. Eng* **SE-5(2)** 128-137
8. Grama A Y, Gupta A and Kumar V 1993 *Isoefficiency: measuring the scalability of parallel algorithms and architectures*, IEEE Par. & Distr. Tech.: Systems & Applications **1(3)** 12-21
9. G Bauer *et al* 2010, Monitoring the CMS Data Acquisition System, *J. Phys.: Conf. Ser.* **219** 022042
10. Birman, K. and Joseph, T., Exploiting virtual synchrony in distributed systems *in Proceedings of the eleventh ACM Symposium on Operating systems principles (SOSP '87)*, 1987. pp. 123-138.
11. Oracle database, http://www.oracle.com
12. SQLite database, http://www.sqlite.org
13. Adobe Flash Builder, http://www.adobe.com/
14. Adobe Flex, http://www.adobe.com
15. Oracle TimesTen In-Memory Database, http://www.oracle.com
16. *Software As A Service: Strategic Backgrounder*. Washington, D.C.: Software & Information Industry Association. 28 February 2001

CONTACTS

Name: Luciano Orsini
Organization: CERN
Email: Luciano.Orsini@cern.ch
Phone: +41227671615
Web: http://xdaq.web.cern.ch

Name: Andrea Petrucci
Organization: CERN
Email: Andrea.Petrucci@cern.ch
Phone: +41227670808
Web: http://xdaq.web.cern.ch

INTRODUCTION

The Compact Muon Solenoid (CMS) [1] is a general-purpose particle detector at the Large Hadron Collider (LHC) [2] at CERN in Geneva, Switzerland. The CMS Data Acquisition (DAQ) [3] is responsible to build and filter events from about 600 data sources at a maximum trigger rate of 100 KHz. The Error and Alarm system has to provide the facilities to retrieve, process, store and display errors and alarms occurred during the operation of the CMS experiment.

The DAQ is composed by few thousands of hosts [4][5] and O(20000) interdependent applications that need to be monitored in near real-time. One of the critical success factors for an error and alarm management is scalability, several dimensions of scaling requirements [6][7][8] (e.g. numerical and geographical) have been considered to cover all aspects of the system design. The main purpose of the CMS alarm and error system is to alert the operator when an abnormal operation occurs, but also to keep history of all past problems for post mortem analysis.

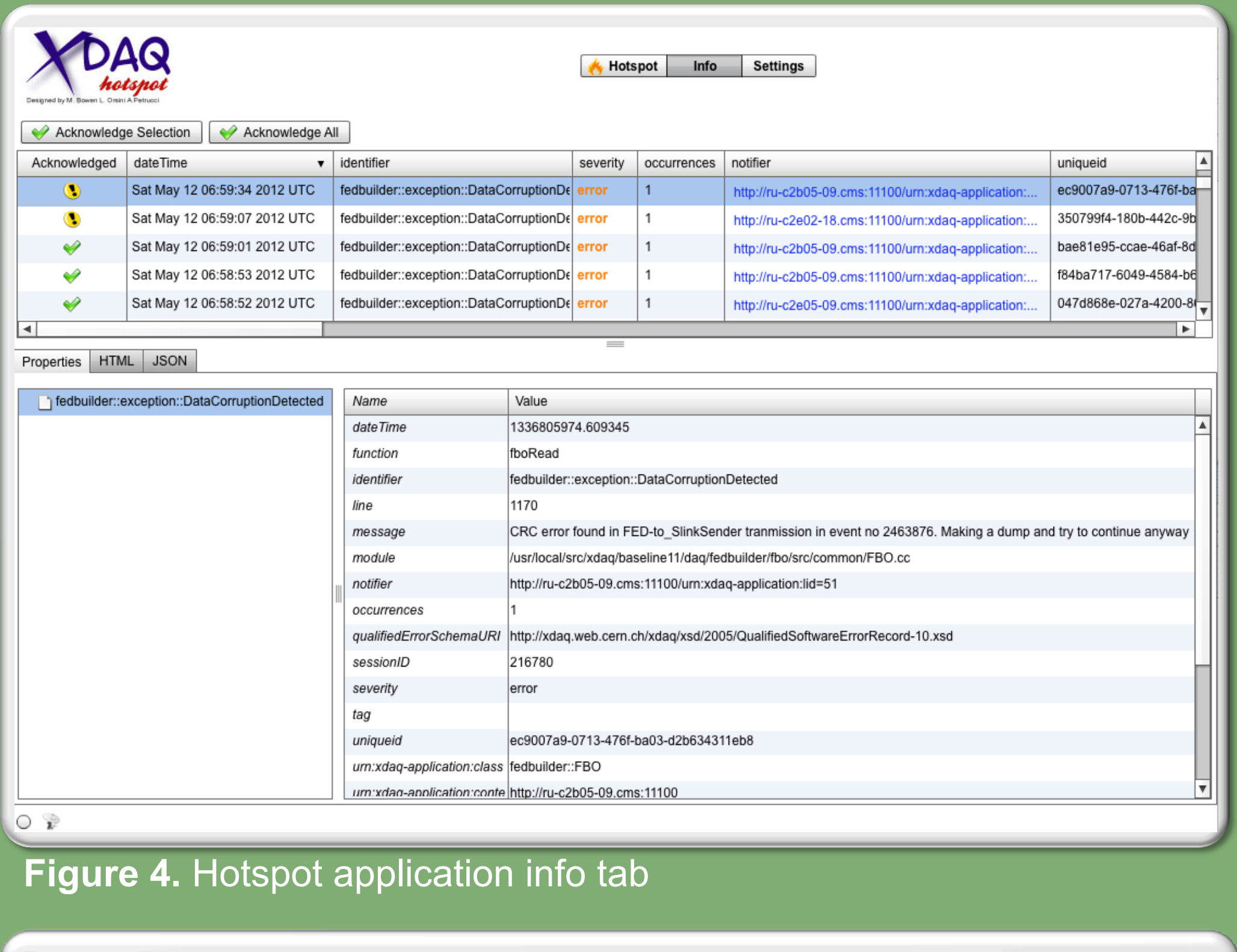
The presented design is based on four powerful concepts: only notify important conditions, notify in time, respond, and provide guidance. The suggested infrastructure fits these needs by providing a set of expandable and reusable solutions allowing use of the alarming system for development, test and operation scenarios.

ARCHITECTURE AND DESIGN

The architecture is based on the XDAQ Monitoring & Alarming Service (XMAS) [9], which provides several plug-ins specialized for specific tasks and services to support and implement a fully scalable distributed monitoring and alarming system. As shown in **Figure 1**, the system builds upon a scalable publisher-subscriber [10] service consisting of a pool of eventing applications orchestrated by a load balancer application (broker). When a DAQ application detects *an anomaly* act as data producers through sentinel services to publish an error or an alarm to the eventing service. The spotlight service is responsible for collecting, processing and storing events. It subscribes to the eventing service specifying categories of error and alarm events it wants to receive. The spotlight supports two databases technologies: Oracle [11] for CMS production system and sqlite [12] for testing and small setups. The visualization components are called Hotspot and Coldspot, both applications are developed using Adobe Flash builder [13]. As shown in **Figure 2**, two different report scenarios can be identified: applications that detect persistent deviations from the normal system behaviour report errors. Deviations may also be transient, meaning an alarm is fired and eventually revoked when the asserted condition is resolved.

VISUALIZATION

In the CMS error and alarm system are available in two different visualization scenarios to support the operation of the CMS experiment: online display and post-mortem display. The CMS operators use the online display during data taking to detect deviations from the normal system behaviour. The CMS experts are able to analyze past errors using the post-mortem display. Hotspot (online display) is an Adobe Flex [14] application used to view and retrieve the error and alarm data stored and retrieved by Spotlight. Hotspot displays errors and alarms according the CMS data acquisition abstract model system. Errors and alarms are associated to elements of the system model and displayed corresponding to their severity levels. The tool offers different views of the model such as tree navigation, heat maps and tables. **The Figure 3**,**4** show the main views of Hotspot, the left-most panel of the main view represents the model tree that categorizes errors into nodes within the tree.



Acknowledged	dateTime	identifier	severity	occurrences	notifier	uniqueid
!	Sat May 12 06:59:34 2012 UTC	fedbuilder::exception::DataCorruptionDe	error	1	http://ru-c2b05-09.cms:11100/urn:xdaq-application:...	ec9007a9-0713-476f-ba
!	Sat May 12 06:59:07 2012 UTC	fedbuilder::exception::DataCorruptionDe	error	1	http://ru-c2e02-18.cms:11100/urn:xdaq-application:...	350799f4-180b-442c-9b
✓	Sat May 12 06:59:01 2012 UTC	fedbuilder::exception::DataCorruptionDe	error	1	http://ru-c2b05-09.cms:11100/urn:xdaq-application:...	bae81e95-ccae-46af-8d
✓	Sat May 12 06:58:53 2012 UTC	fedbuilder::exception::DataCorruptionDe	error	1	http://ru-c2b05-09.cms:11100/urn:xdaq-application:...	f84ba717-6049-4584-b6
✓	Sat May 12 06:58:52 2012 UTC	fedbuilder::exception::DataCorruptionDe	error	1	http://ru-c2e05-09.cms:11100/urn:xdaq-application:...	047d868e-027a-4200-8

fedbuilder::exception::DataCorruptionDetected

Name	Value
dateTime	1336805974.609345
function	fboRead
identifier	fedbuilder::exception::DataCorruptionDetected
line	1170
message	CRC error found in FED-to_SlinkSender tranmission in event no 2463876. Making a dump and try to continue anyway
module	/usr/local/src/xdaq/baseline11/daq/fedbuilder/fbo/src/common/FBO.cc
notifier	http://ru-c2b05-09.cms:11100/urn:xdaq-application:lid=51
occurrences	1
qualifiedErrorSchemaURI	http://xdaq.web.cern.ch/xdaq/xsd/2005/QualifiedSoftwareErrorRecord-10.xsd
sessionID	216780
severity	error
tag	
uniqueid	ec9007a9-0713-476f-ba03-d2b634311eb8
urn:xdaq-application:class	fedbuilder::FBO
urn:xdaq-application:conte	http://ru-c2b05-09.cms:11100

Figure 4. Hotspot application info tab

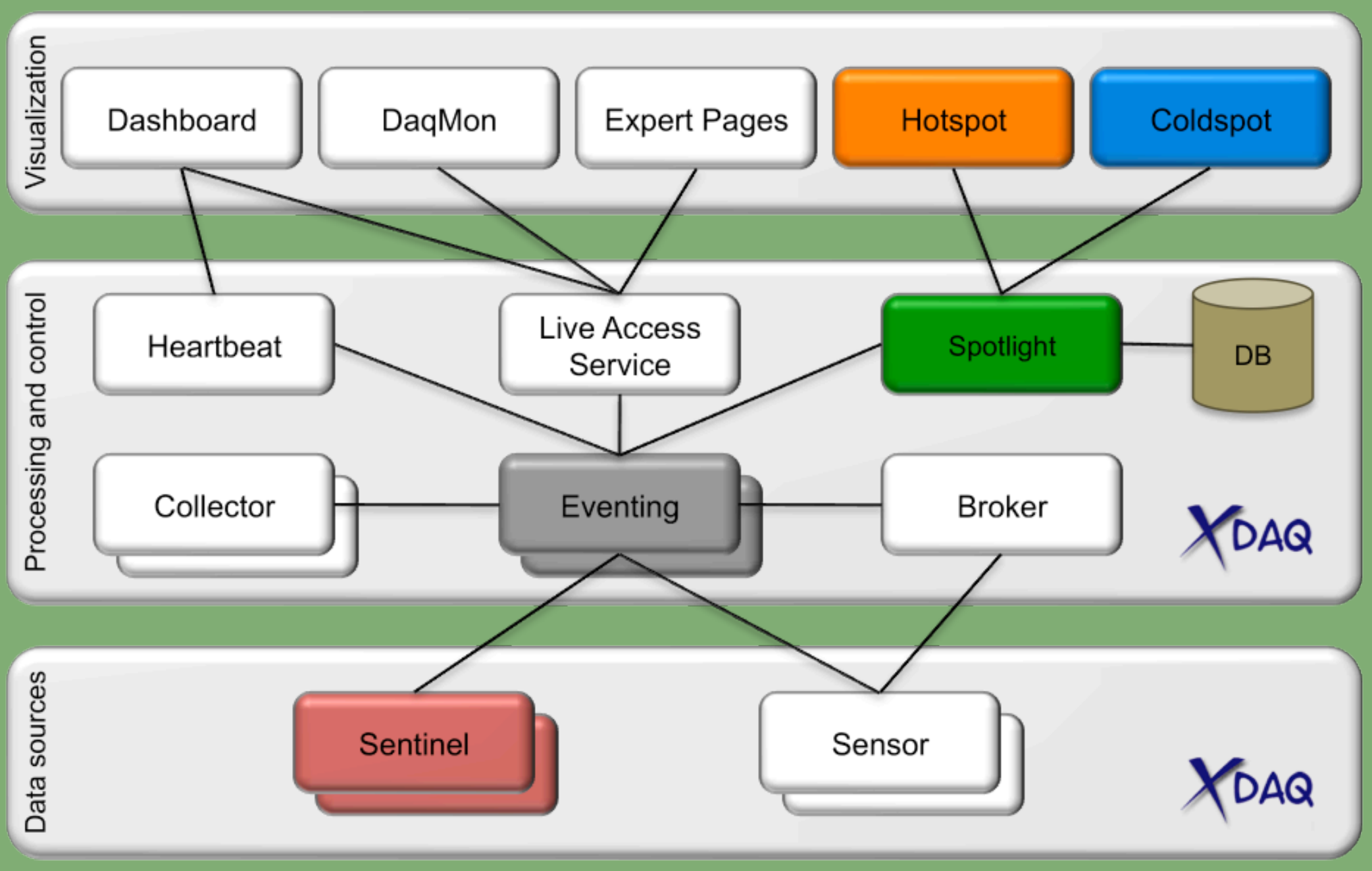
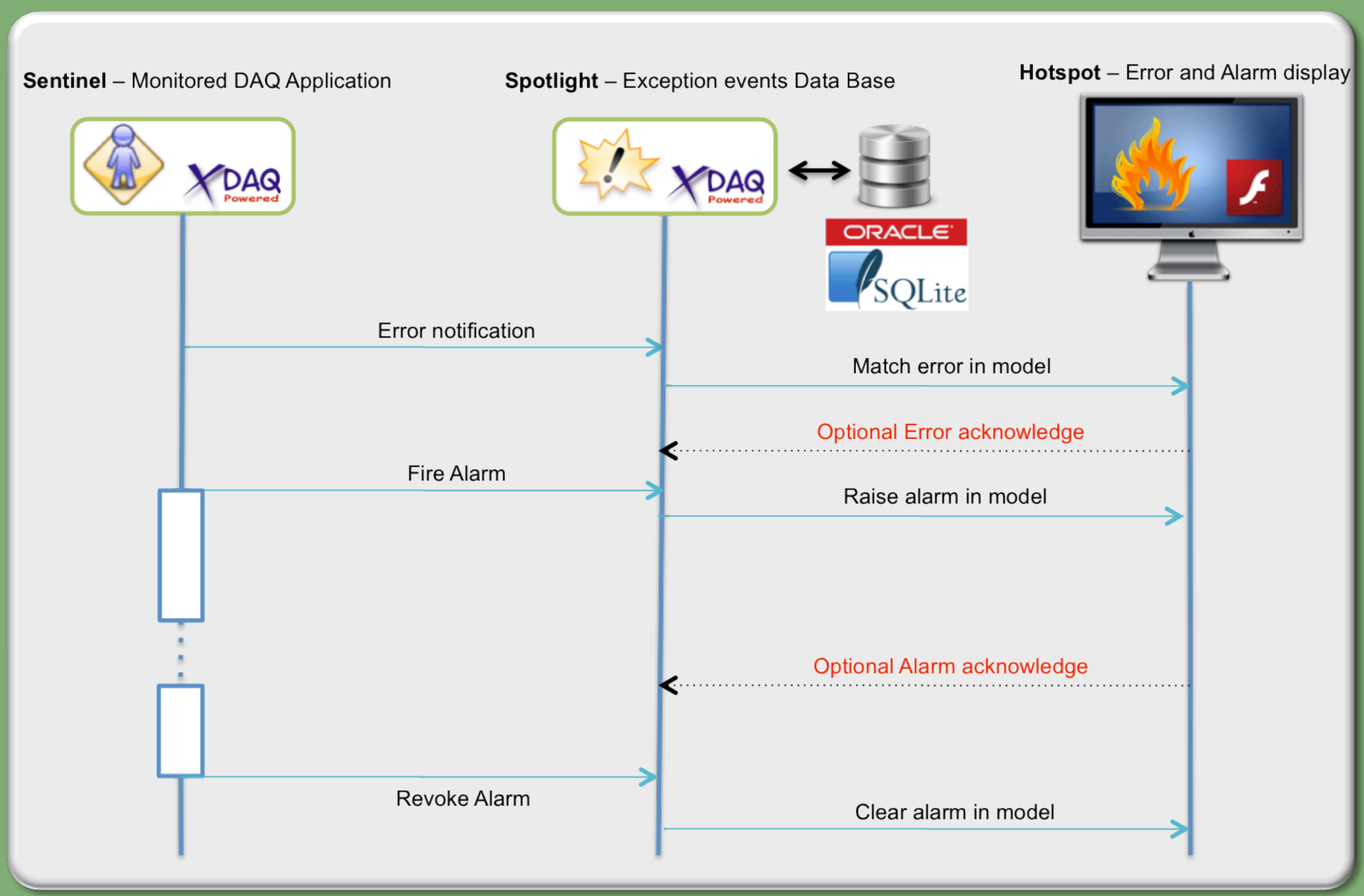


Figure 1. In colour the element involved in the reporting, recording and display of errors and alarms.



Sentinel – Monitored DAQ Application
Spotlight – Exception events Data Base
Hotspot – Error and Alarm display
XDAQ Powered
XDAQ Powered
ORACLE
SQLite
Error notification
Match error in model
Optional Error acknowledge
Fire Alarm
Raise alarm in model
Optional Alarm acknowledge
Revoke Alarm
Clear alarm in model

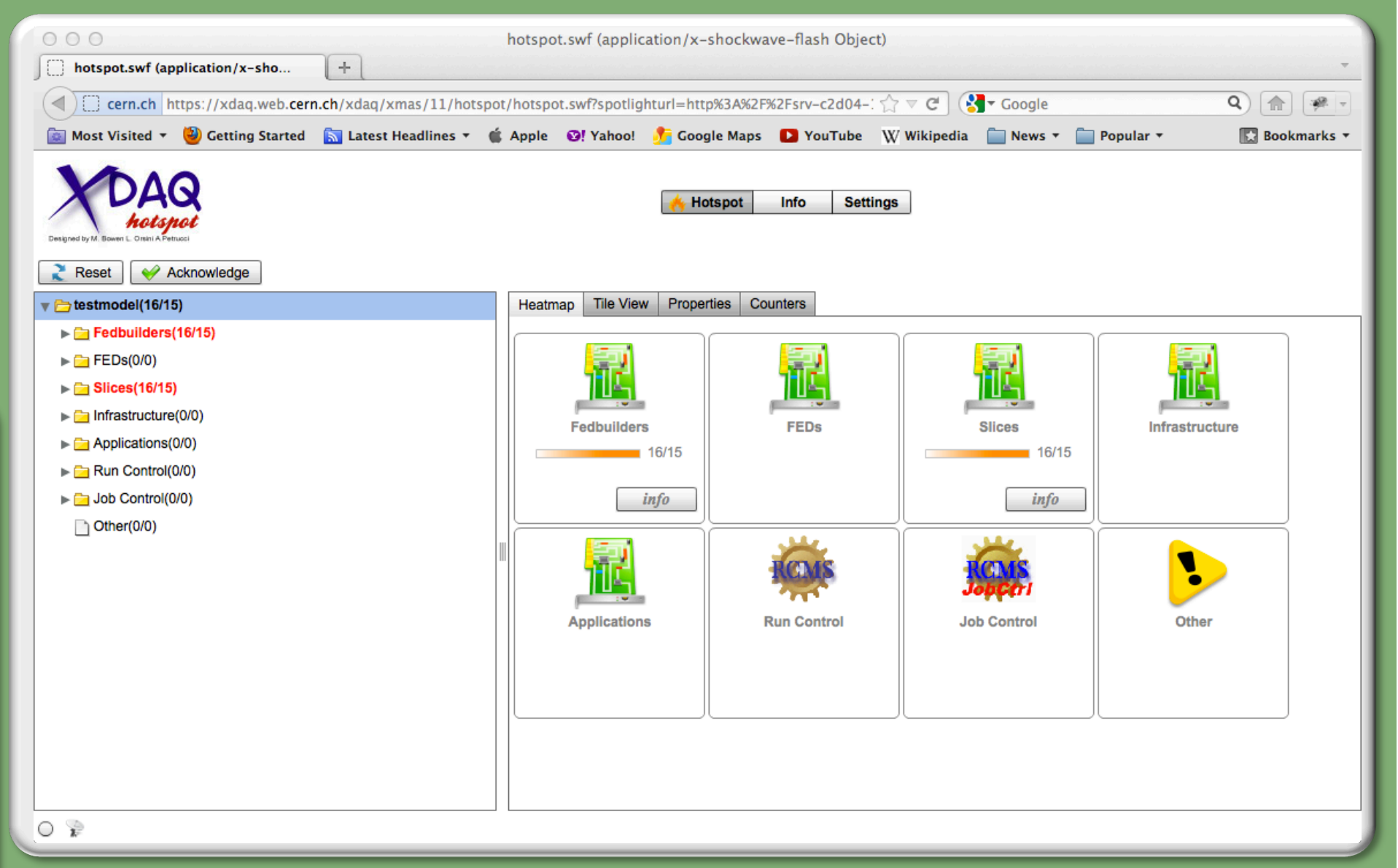
Figure 2. Error and Alarm interaction diagram.

BENCHMARKS

The maximum performance achieved for the CMS error and alarm system depends on the database capabilities. SpotlightOCCI has been measured to be capable to perform at approximately 600Hz (insertions per second). Using with a TimesTen memory cache, spotlightTT was able to perform approximately at 2kHz. Spotlight2g, for comparison, stored approximately 1kHz insertions per second.

PERSISTENCY

The spotlight application is responsible for the storage and retrieval of error and alarm data. Three spotlight applications have been implemented to support different requirements: *Spotlight2g*, utilized a database called SQLite to store the exceptions. SQLite is a in-process library that implements a self-contained, server-less, zero-configuration and transactional SQL database engine. *SpotlightOCCI*, making use of Oracle's OCCI API, works to store this exception data in an Oracle database. *SpotlightTT,* utilized an in memory database called Oracle TimesTen [15]. This database is designed for low latency, high-volume data, event and transaction management.



hotspot.swf (application/x-shockwave-flash Object)

https://xdaq.web.cern.ch/xdaq/xmas/11/hotspot/hotspot.swf?spotlighturl=http%3A%2F%2Fsrv-c2d04-

- testmodel(16/15)
  - Fedbuilders(16/15)
  - FEDs(0/0)
  - Slices(16/15)
  - Infrastructure(0/0)
  - Applications(0/0)
  - Run Control(0/0)
  - Job Control(0/0)
  - Other(0/0)

Figure 3. Hotspot application main view

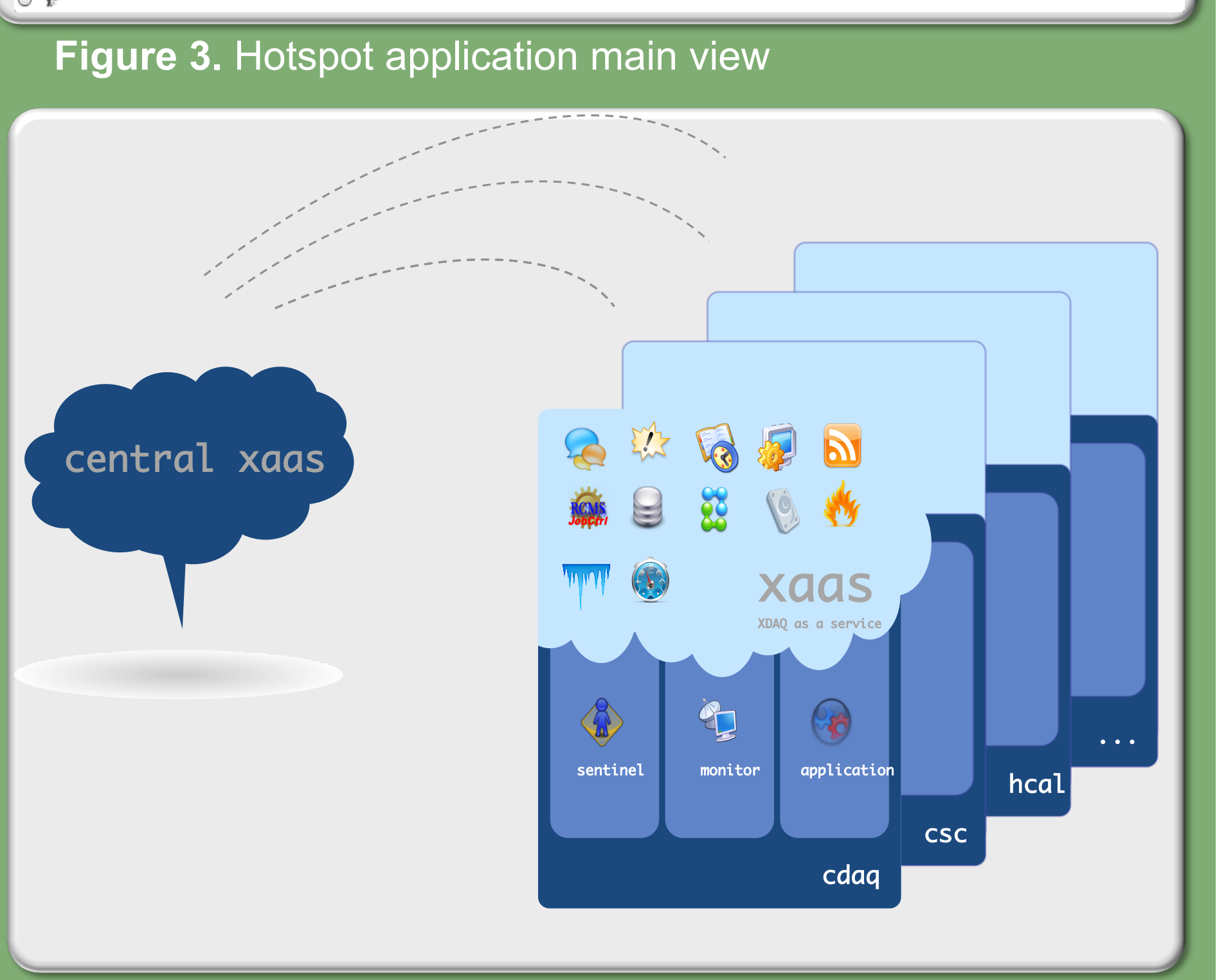


Figure 5. Central Xdaq as a Service architecture

INTEGRATION

The CMS sub-detectors use the XDAQ framework to develop the online software needed for the data taking, and are responsible to bring data from their sub-detector front-end system to front-end drivers. Different teams are involved in developing the sub-systems software and alternative approaches are taken for the same problem. One example is the error reporting: storing errors in a local disk, collecting logs in a central place etc. The central DAQ group aims to standardize the error and alarm management for all CMS online system. To achieve this, a common software framework is not enough, but a common infrastructure is needed. To cope with this problem the Xdaq as a Service (XaaS) has been designed based on the software as a service concept [16]. XaaS is a full set of interoperable services that provide standard functionalities for use in XDAQ environment. Each CMS sub-detectors should have a XaaS infrastructure maintained by the central DAQ team and a central XaaS system could collect error and alarm from all the CMS experiment as shown in **Figure 5**.