• Expert Systems in ATLAS TDAQ
  – Why?
  – From Run 1 to Run 2
• Introducing a Complex Event Processing engine
  – ESPER from EsperTech
• Intelligent systems in Run 2
  – The Central Hint and Information Processor (CHIP)
  – The Shifter Assistant
• A different point of view
  – The Shifter Assistant Replay (SAReplay)
• Conclusions and outlook
Why?

Automation & Error Management

Maximize system efficiency

- Reduce human interventions
- Dealing fast and effectively with errors and failures
- Effective analysis and continuous monitoring of the system

Optimize man power

- A way to formalize expert’s knowledge
- Reduce the load on the operator
From Run 1 to Run 2

- **Online Recovery**
  - Based on CLIPS

- **Shifter Assistant (SA)**
  - Tool to assist the shifter in accomplishing his/her duties
  - Based on a Complex Event Processing (CEP) engine, ESPER

- **Central Hint and Information Processor (CHIP)**
  - Replaces Online Recovery
  - Same CEP engine as the SA
Complex Event Processing

• A set of technologies to process events and discover complex patterns among streams of events
  – Used in financial analysis, wireless sensor networks, business process management

• A cross between Data Base Management System and Rule Engines

• Main characteristics
  – Continuous stream processing
  – Support for time/size windows, aggregation and grouping events
  – SQL-like pattern languages
    • Augmented with constructs to express event relationships (time, cause and aggregation)
    • Streams replacing tables in a continuous evaluation model
A CEP Engine - ESPER

- **Java based**
  - Events as Java beans, XML documents, classes or key-value pairs
  - Fully embeddable
  - Open source

- **Support for advanced stream analysis**
  - Correlation, aggregation, sliding windows, temporal patterns

- **Knowledge base expressed in the Event Processing Language (EPL)**
  - Rich SQL-like language to express complex queries

- **Natively high-configurable multi-threaded architecture**
  - Inbound and outbound thread pools, timers

- **Support for historical data**
  - Full control over time!

- **Built-in advanced metrics**

Not available in CLIPS
Detecting Patterns

- Detect specific events
  - i.e. React on FATAL or ERROR messages from some applications

- Detect pattern on time windows
  - 5 min
  - i.e. React if a component is reporting high load conditions for more than 60 seconds

- Detect absence of events
  - i.e. React if ATLAS is not running but physics collisions are provided by the accelerator

- Elaborate complex results
  - i.e. Detect unbalanced rates to the data storage
CEP in ATLAS DAQ

ESPER

CHIP

Able to detect failures and perform automatic recovery procedures

Shifter Assistant

Dealing effectively with problems requiring the human intervention
CEP in ATLAS DAQ

How it works

Information gathering

Patterns

Conditions

Information processing
(Knowledge Base)

Recoveries and procedures

Actions

Operational monitoring

Application messages

Run Control

TDAQ system largely deterministic → Possible to identify “signatures” and react properly
## Data Sources

<table>
<thead>
<tr>
<th>Typical information sources</th>
<th>Run Control</th>
<th>Application Messages</th>
<th>Operational Data</th>
<th>System Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Process status</td>
<td>Different severities</td>
<td>LHC status</td>
<td>Enabled detectors</td>
</tr>
<tr>
<td></td>
<td>Executed Commands</td>
<td>Reporting anomalies</td>
<td>Detector working parameters</td>
<td>Application parameters</td>
</tr>
<tr>
<td></td>
<td>FSM states</td>
<td>Can trigger on-demand actions</td>
<td>Run parameters</td>
<td>Application hierarchy</td>
</tr>
</tbody>
</table>

### Diagram
- **Event reception/retrieval**
- **Pre-processing (Java objects)**
- **Injection**
CHIP and the Run Control

- Applications in the Run Control (RC) are organized as a hierarchical tree
  - “Controllers” are responsible of “leaf applications”
- CHIP is the “brain” of the RC system
  - Error management
  - Anomaly detection
- Controllers and CHIP communicate through a well defined interface
CHIP: Recoveries and Automation

Core
- Run Control error management
- Dealing with application failures

(Stop-less) Recovery
- (Re)synchronization of detectors
- Removal and re-insertion of busy channels
- Full reconfiguration of detectors

Automation
- Setting ATLAS reference clock
- Moving to physics mode with stable beams
- Detector specific procedures

Not The Full List!
More than 300 rules in 26 different contexts
CHIP & EPL: An Example

<table>
<thead>
<tr>
<th>Application</th>
<th>Issued</th>
<th>Severity</th>
<th>Msg Id</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCM:HLT-12:tpu-rack-1...</td>
<td>13 Mar 2016 18:02:09 CET</td>
<td>Fatal</td>
<td>rc:TransitionFailed</td>
<td>The transition “CONNECT” has not been properly co...</td>
</tr>
</tbody>
</table>

### Problem detection - Application sending a FATAL error

```sql
@Name('INSERT INTO_Problem_ERSFatal_New')
on ERSEvent(severity in ("FATAL", "Fatal")) as ers
insert into Problem
select
crcAppTable.controller,
ers.applicationName,
Problem$TYPE.ERS_FATAL,
Problem$STATUS._NEW,
Problem$ACTION.NONE
from RCApplicationTable as rcAppTable
where ers.applicationName = rcAppTable.name
and rcAppTable.isController = true;
```

### Action decision - Set error state

```sql
@Name('INSERT INTO_Problem_ERSFatal New')
on Problem(type=Problem$TYPE.ERS_FATAL, status=Problem$STATUS._NEW, action=Problem$ACTION.NONE) as p
insert into Problem(controller, application, type, status, action)
select p.application as controller, p.application as application, p.type, p.status, Problem$ACTION.SET_ERROR;
```

### Action execution - Call executor to send command

```sql
@Name('SUBSCRIBER_ProblemExecutor2')
@Subscriber(className='chip.subscriber.core.ProblemExecutorSynch')
select * from ProblemExecutor(action in (Problem$ACTION.SET_ERROR, Problem$ACTION.REMOVE_ERROR));
```
CHIP & EPL: More Complex Example

```sql
@Name('INSERT INTO Problem_BadHost_Resolved')
@Hint('PREFER_MERGE_Join')
context SegmentedByProblemPerApplication
insert into Problem
select
p.controller,
p.application,
Problem$TYPE.BAD_HOST,
Problem$STATUS.RESOLVED,
Problem$ACTION.NONE
from
pattern [ every p = Problem
  type = Problem$TYPE.BAD_HOST, status = Problem$STATUS.WAIT_FOR_RESOLVED ) ->
  ( every t = TestFollowUp
    ( applicationName = p.application,
      globalTestResult = TestResult.PASSED,
      compTestResult.testResult = TestResult.PASSED,
      status = TestFollowUp$STATUS.NEW,
      action = TestFollowUp$ACTION.NONE )
  )
  or
  ( every app = RCAApplication
    name = p.application )
) and not Problem( application = p.application,
  type = Problem$TYPE.BAD_HOST,
  status in ( Problem$STATUS.RESOLVED, Problem$STATUS.DONE ) ) as pat
where
pat.t.component is
(select runningHost from RCAApplicationTable where name = pat.p.application)
or
app.status is not STATUS.UP;
```
CHIP: Metrics Analysis

• Exploiting ESPER’s built-in metrics
  – Detailed information for every single rule in the knowledge base

• Real-time and historical data
  – Leveraging the flexibility of P-Beast (see Igor’s talk on Thursday) and Grafana

• “Live” enabling/disabling and configuration
CHIP: Performances

- Standard ATLAS configuration
- DAQ system cycled through various states
- Provoked various failures

Average Execution Time per Statement

- INSERT INTO LHCEvent_SB
- INSERT INTO ISEvent_Pixel1
- Print_Errors_On_IGNORE_ERROR
- INSERT INTO Action_HITSv_Dead_New
- SUBSCRIBER_Clockswitch
- INSERT INTO LHC_safe
- CREATE_WINDOW_LHCEventsWindow
- INSERT INTO Action_AppDead_New
- getAllSubscriptions
- getAllStopless_Ers
- SUBSCRIBER_HeartRecoveryExecutor
- SUBSCRIBER_ProblemExecutor2
- INSERT INTO WarmStart_WarmStop_New_1
- getAllAppDead
- INSERT INTO ClockSwitch_SelectInternal_RisRunning_New
- a50c3332-1025-401e-9c81-64b016a752d
- INSERT INTO Problem_AppFailed_New
- INSERT INTO Problem_AppError_New
- INSERT INTO Stopless_Removal_New
- INSERT INTO Stopless_Recovery_New
- CREATE_WINDOW_HIREcoveryTable
- INSERT INTO HIersRecovery_Fork
- INSERT INTO HIRecovery
- INSERT INTO Controller
- INSERT INTO RCAppCommandChangeState
- INSERT INTO ErrorTable
- INSERT INTO HIRecovery_Stop
- INSERT INTO Problem_BadHost_Resolved

3 μs average* (weighted on # executions)

* Running on Intel Xeon E5-2680 V3 - 64 GB RAM
The Shifter Assistant

What

• A tool meant at guiding the operator in his daily work
• Diagnosing problematic situations and suggesting action to take
• Remind the operator he should (not) do something

Aim

• Reduce and simplify shifter tasks
• Help shifters with more detailed and pertinent information
• Be more efficient, avoid repetition
• Promptly notify operators of problems and failures
The Shifter Assistant: Web Interface

- **Context**
- **Severity**
- **What to do**
Leveraging ESPER’s support for import of historical data to build a tool to validate the Shifter Assistant knowledge base.

Facilitate sharing of knowledge, safer and easier integration of new rules.

Online

Persistency

Offline Data Replay

Information Servers

Application Messages

P-BEAST

SQL DB

Archived System Configuration

Merging & Sorting

Shifter Assistant

New Rules

Leveraging ESPER’s support for import of historical data to build a tool to validate the Shifter Assistant knowledge base.

Facilitate sharing of knowledge, safer and easier integration of new rules.

Online

Persistency

Offline Data Replay
Going Further: Neural Networks

Beyond static definitions of rules

A Different Approach

• Teach the NN what a “normal” system behavior is
• Identify abnormal situations independently of the parameter space
  – Classification of anomalies

A Typical Fill Profile

Boxes represent anomalies the NN algorithm is able to detect
• In both Run 1 and Run 2 the use of Expert Systems for automation and recovery proved to be a valuable asset
  – Reduce probability of mistakes
  – Improve latency (computers are faster than humans...)
• The introduction of a CEP engine has added flexibility and simplification
  – Improved anomaly detection
  – New complex patterns
  – Re-use of historical data
  – Advanced configuration
• Looking forward for more advanced automation and anomaly detection