



Rule-based Systems for Early Warning

CHORIST Seminar,
Istanbul, 23-4 June 2009

Introduction

- What is a rule-based system?
 - What do I mean
- Why is it suitable for use in early warning systems?
 - Advantages and disadvantages
- How can it be used?
 - examples

CHORIST Design Objectives

- Configurability
 - for different authorities
 - for different seasons
 - for changing circumstances

Without rewriting the software
- Modularity
 - In/exclude parts for different customers

What is a Rule-based System?

- Rule-based programming attempts to derive execution instructions from a starting set of data and rules, which is a more indirect method than using a programming language which lists execution steps straightforwardly.
- It is these rules, which can be changed outside the software, to address configurability

Constituents of a Rule-based System

- A rule based system has:
 - a set of rules, outside the program, often written in a special but understandable syntax
 - an engine, inside the program, that can read and interpret the rules
 - Efficient in-memory storage and a sophisticated algorithm for efficient processing of rules

A Database “turned inside out”

- triggers actions when event conditions occurs among event streams - which can be thought of as a database turned upside down where statements are registered and data streams flow through.
- For efficiency and scalability, e.g. performance figures, with just a 2 CPU dual core 3GHz platform, Esper processes more than 500 000 event/s with a latency average below 3 microseconds, and below 10 microseconds with more than 99% predictability.

Terminology

- Rule-based
- Complex Event Processing (CEP)
 - much about performance and scalability
- Optimisation Tools
 - E.g. minimise ...
 - can be a part but less real time
- Event stream processing (ESP)
- Event correlation engine (CEP)

A Simple Approach

```
Receive sensor reading;  
  
Extract sensor_id;  
Extract sensor_value;  
  
If (sensor_id = 1289)  
    if (sensor_value > 5)  
        Issue alarm  
    endif  
Elseif (sensor_id = 1290)  
  
Elseif ...  
  
Endif
```

- “Hard-coded”
- Easy to understand
- Difficult to change
- Not flexible
- ... But it works

A Better Approach

```
For all values in DB
  read threshold $t;
  read sensor_id $i

Receive sensor reading;
Extract sensor_id;
Extract sensor_value;

For all values in DB
  If (sensor_id = $i)
    if (sensor_value > $t)
      Issue alarm
    Endif
  Endif
Endfor
```

- More flexible
- Can change by updating database without changing code
- etc

A Sophisticated Approach

```
Initialise RuleEngine;  
Read rules from Database;  
For each rule  
    RuleEngine.add (rule)  
  
RuleAction:  
    Issue alarm
```

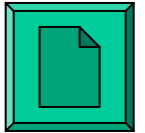
- Very flexible
- Less easy to understand
- Can change rules
- Can accommodate more rule types as we will see
- Needs rule language and syntax

Tools and Characteristics

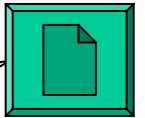
- Wikipedia lists 15 rules engines, e.g.
 - Drools, Esper, Mandarax, Versata
- Commercial / Open Source
- Well supported / less well supported
- Some, e.g. ILOG, cater for optimisation
 - Calculate something optimally within constraints (rules)
 - Tend to be less real time
 - Can use different models , e.g. linear programming

Rules Examples – Beyond Simple Thresholds

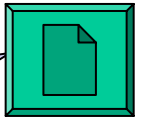
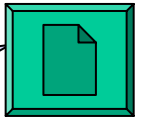
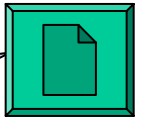
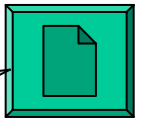
```
select count(*) from Event.win:time_batch(1 second)
```



```
select * from pattern [every EventA -> (timer:interval(10 sec)  
and not EventB)]
```



```
select Event1.num as N1, Event2.num as N2 Event1.win:time(30 sec) ,  
Event2.win:time(30 sec), where Event1.id = Event2.id
```



More Rule Examples

- How do I keep a separate window of events per category and compute aggregates for each category's window?
- How do I use results of one statement in another statement?
- How do I reduce the rate of event output by my statement? How do I get frequent but not continuous results?
- How do I delay data? How do I compare against previous events?
- How do I detect the absence of an event and the presence of an event arriving too late?
- How do I report at a regular interval without any incoming events?
- How do I find missing events arriving in 2 or more streams that are correlated?
- How do I look into the past and consider missing events? How do I do a snapshot, fire-and-forget or on-demand query?

Esper Tool Features

- Tailored SQL-like query language using *insert into*, *select*, *from*, *where*, *group-by*, *having* and *order-by* clauses
- Inner-joins and outer joins (left, right, full) of an unlimited number of streams or windows
- Subqueries including *exists* and *in*
- Sliding windows: time, length, sorted, accumulating, time-ordering, externally-timed (value-based windowing)
- Tumbling windows: time, length and multi-policy; first-event
- Grouping, aggregation, sorting, filtering, merging, splitting or duplicating of event streams

Esper Tool Features

- Output rate limiting and stabilizing, snapshot output
- Named windows
 - Explicit sharing of data windows between statements
 - Multiple and custom entry and exit criteria for events
 - Support for predefined query execution optimized by indexed access, via on-select
- Logical and temporal event correlation
- Crontab-like timer *'at'* operator
- Lifecycle of pattern can be controlled by timer and via operators, repeat-number and repeat-until, every-distinct
- Pattern-matched events provided to listeners

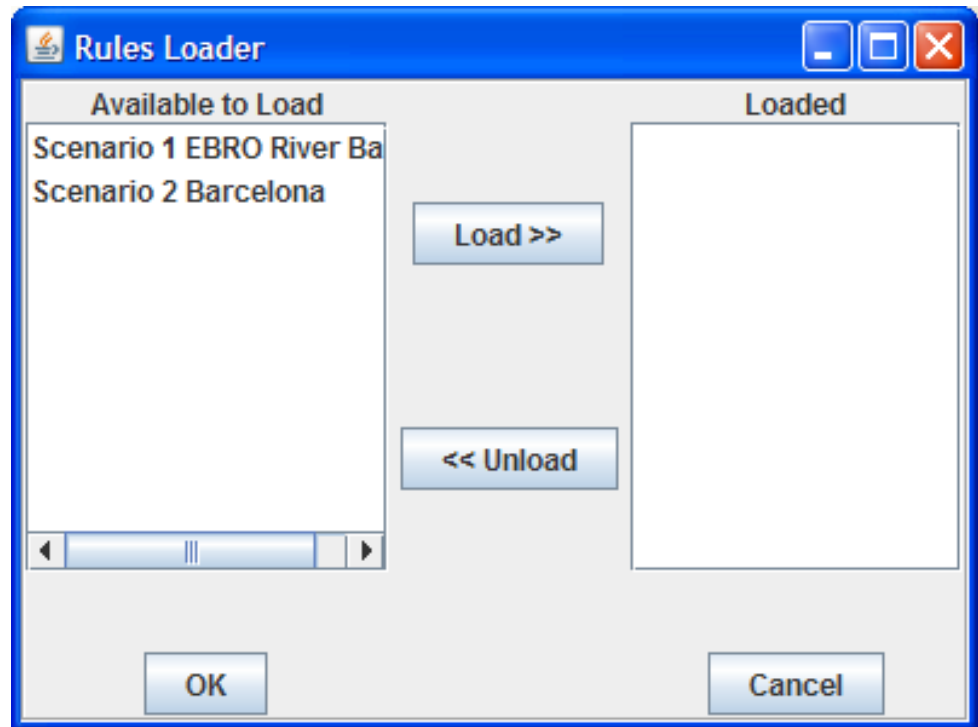
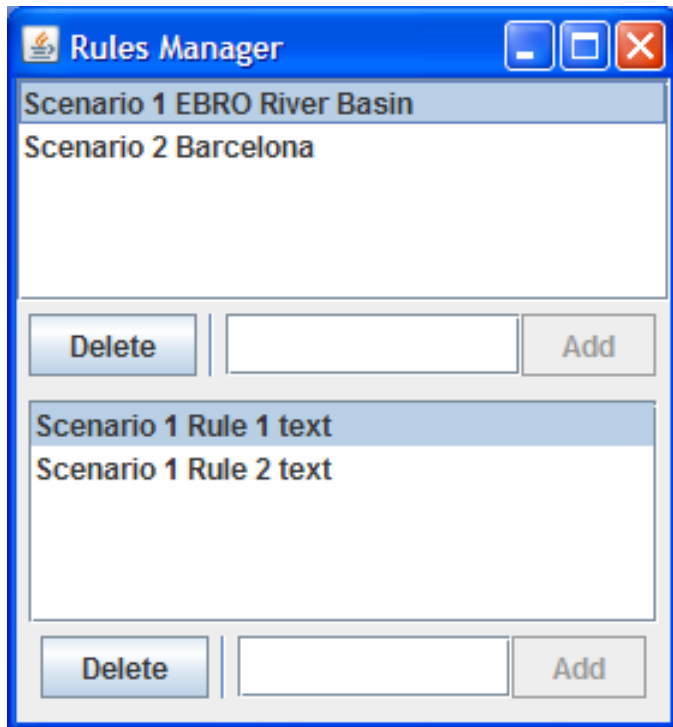
A Tradeoff – Flexibility c.f. Complexity

Manual frequency allocation by skilled personnel

Key		Global	SC	SE	Start	600000	1525
		NW	NC	S	Step	50	0.00125
		NE	S1	S2	X = left		Y = right

Band	IOR		AORE		POR		AORW	
	X	Y	X	Y	X	Y	X	Y
600000	1525.00000							
600050	1525.00125							
600100	1525.00250							
600150	1525.00375							
600200	1525.00500							
600250	1525.00625							
600300	1525.00750							
600350	1525.00875							
600400	1525.01000							
600450	1525.01125							
600500	1525.01250	P_LESV3_CL/P_MESV3_LC					P_LESV3_CL/P_MESV3_LC	
600550	1525.01375							
600600	1525.01500							
600650	1525.01625							
600700	1525.01750	P_LESV3_CL/P_MESV3_LC					P_LESV3_CL/P_MESV3_LC	
600750	1525.01875							
600800	1525.02000							
600850	1525.02125							
600900	1525.02250	P_LESV3_CL/P_MESV3_LC					P_LESV3_CL/P_MESV3_LC	
600950	1525.02375							
601000	1525.02500							
601050	1525.02625							
601100	1525.02750							
601150	1525.02875							
601200	1525.03000							

A CHORIST Rules Manager



- Note the SQL-like language for rules definition
- Behind this is a simple database for storing scenarios and related rules

Conclusion

There is a range of special (and less well known) software tools that can be used in the Early Warning Systems

- The number of these tools is large
- They have a wide range of functional possibilities
- They can respond to the type of environment required by EWS, e.g. performance
- They are well worth investigating

The End

- Any Questions?