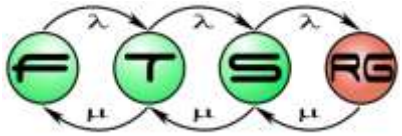
Complex Event Processing
A brief overview

Dávid István
davidi@inf.mit.bme.hu



Key concepts

- **Event**
  - „An immutable record of a past occurrence of an action or state change.”
  - An observable entity triggered by a component of the system (IT infrastructure, software, etc).
- **Event stream**
  - A time ordered sequence of events in time.
- ***Atomic*** event vs. ***Complex*** event
  - A complex event is a compound structure of several atomic events.

Typical use-cases of CEP

- Stock market trading
  - Efficient exploitation of the window-of-opportunity
- IT infrastructure monitoring
  - Evaluating compound metrics
- Security
  - Prevention of dDOS attacks
- Online fraud detection
  - UEFA online betting system
- Automotive vehicle architectures
  - Timing constraint modeling for ECU signals

Processing modes:
- Low-latency online processing
- Batch processing

A slightly different use-case

"Person of Interest" *(original title)*

TV Series - 43 min - Action | Drama | Mystery

8.3

Your rating:

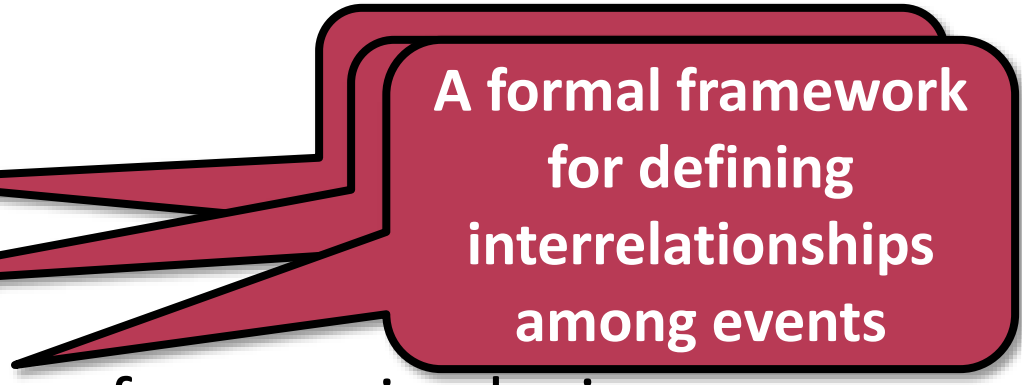
Ratings: **8.3**/10 from 29,071 users

Reviews: 57 user | 10 critic

*„a computer system which uses **information gleaned from omnipresent surveillance** to predict future terrorist attacks and ordinary crimes”*

Background

- Employed techniques
  - Event abstraction
  - Pattern matching
  - Algebraic representation of processing logic (a.k.a. the process algebra)
  - …
- Assumptions and requirements
  - An observable event stream
  - A CEP engine which includes the proper implementation of the processing algebra
  - Scalable behavior (because of the high-volume data)



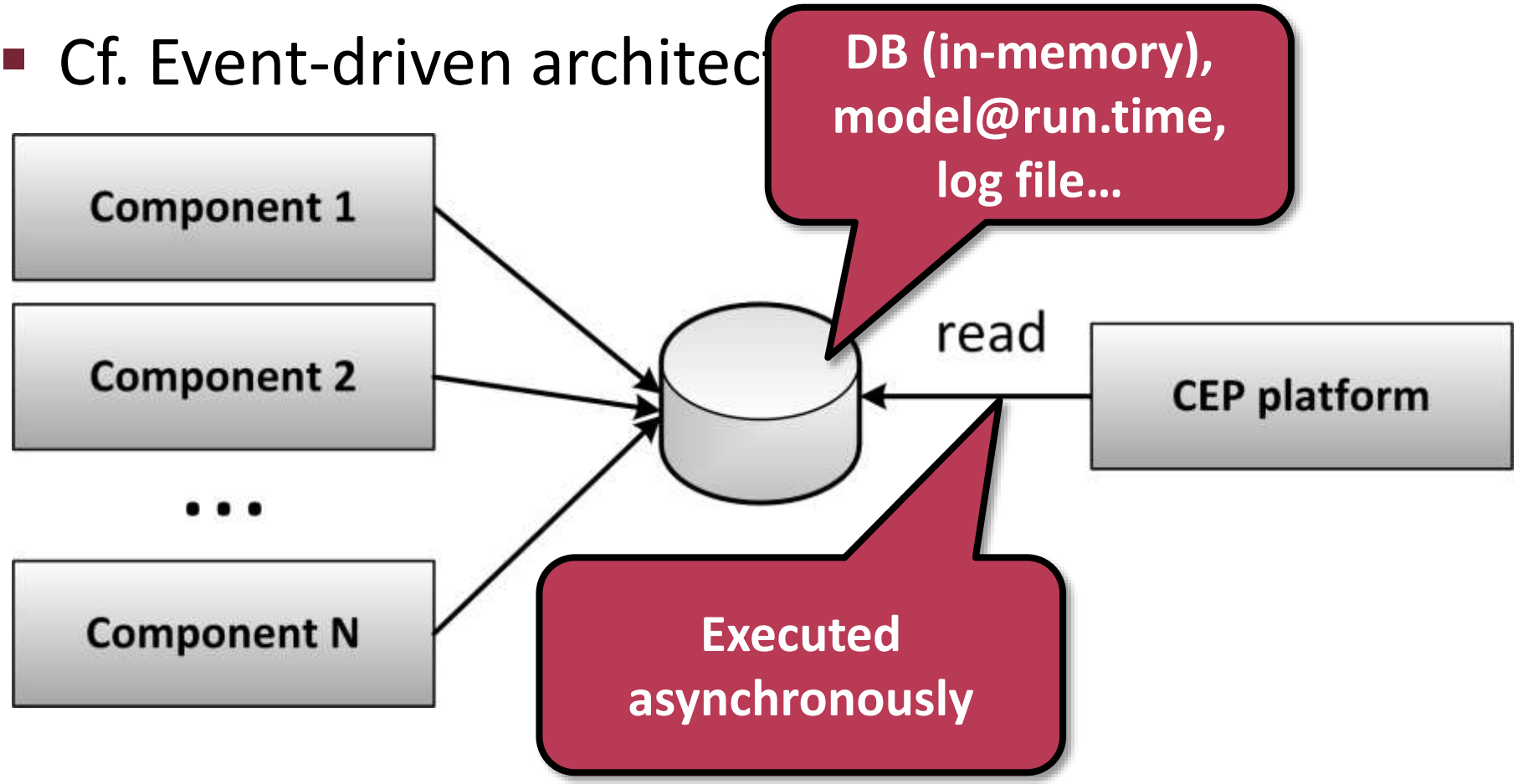
Custom processing algebra?

Go ahead!

Carlson: An Intuitive and Resource-Efficient Event Detection Algebra
Broy, Stefanescu: The Algebra Of Stream Processing Functions
De Lathauwer: Signal Processing Based on Multilinear Algebra

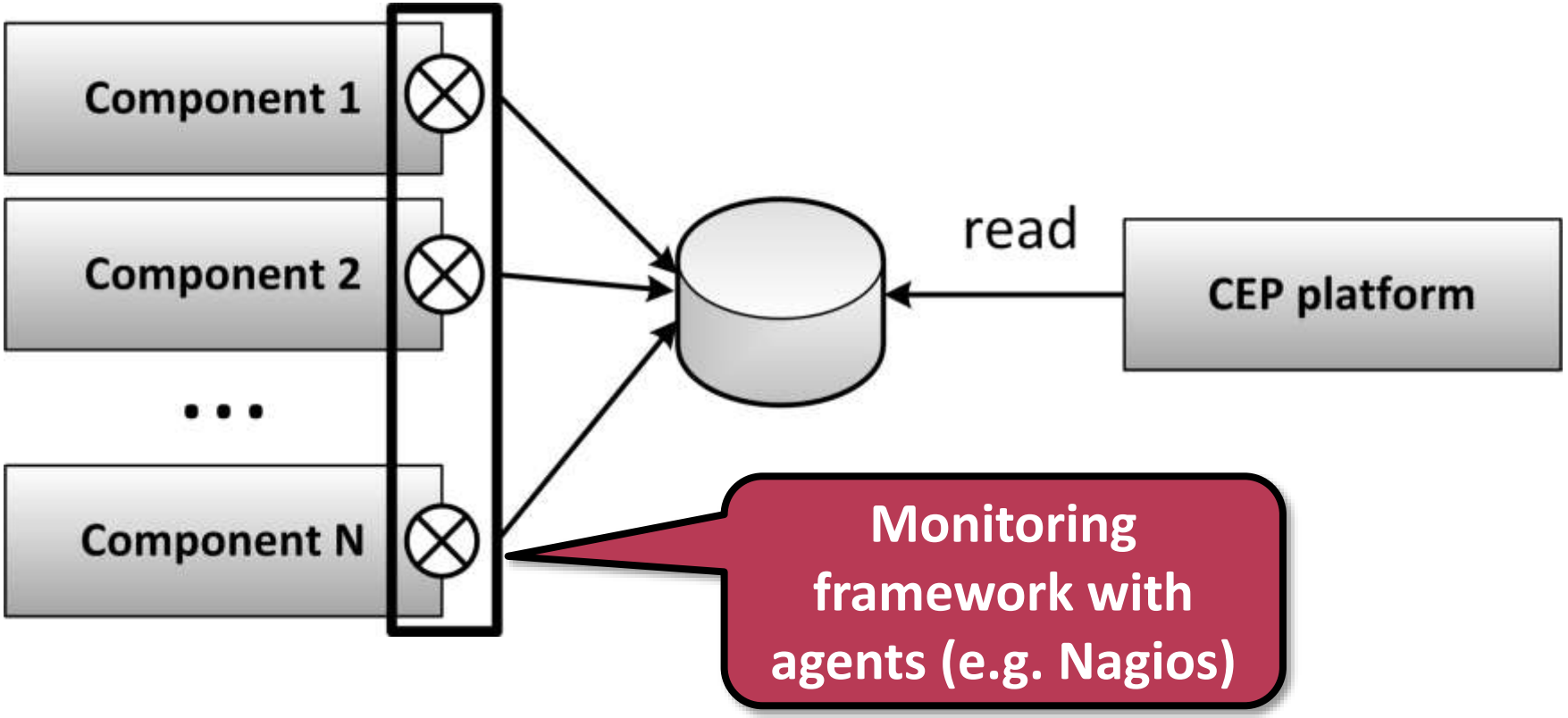
Infrastructure alternatives

- Components themselves publish events into a central repository
- Cf. Event-driven architec



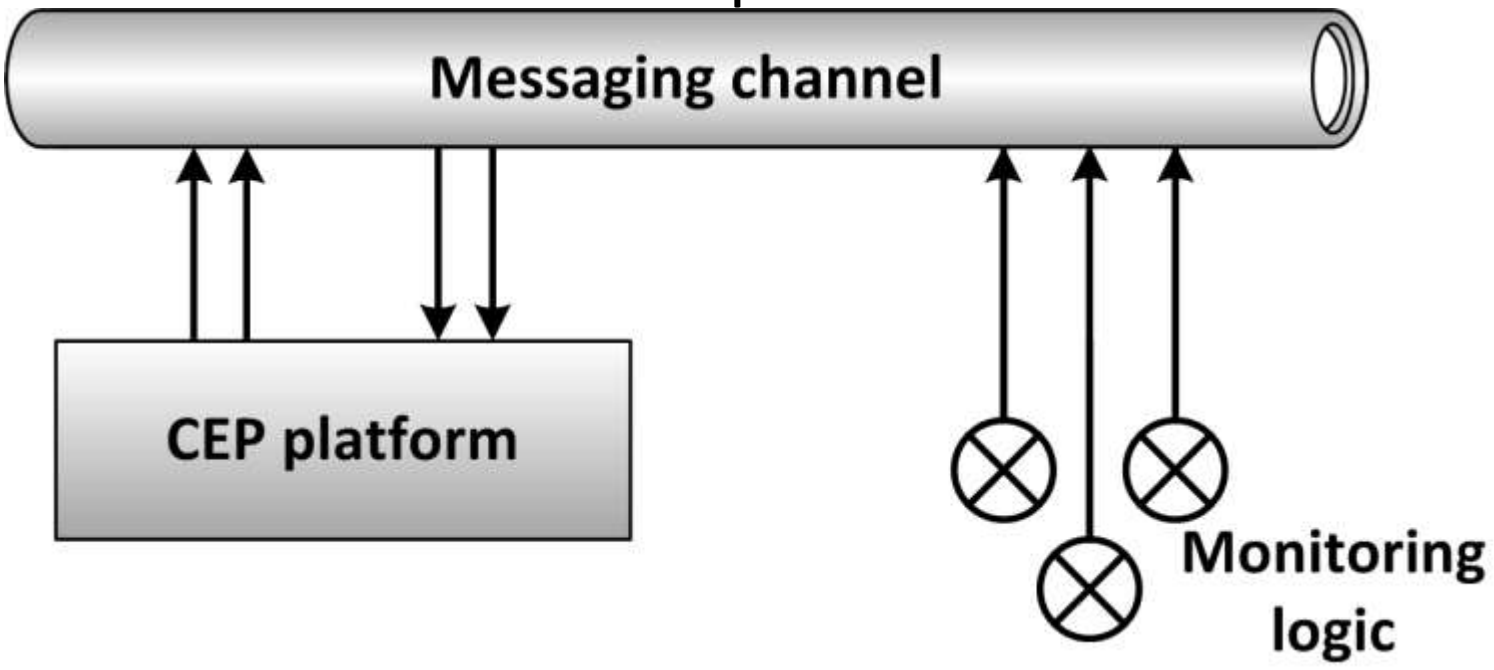
Infrastructure alternatives

- A monitoring framework is employed to collect events from components (instrumentation)



Infrastructure alternatives

- In general: we assume a messaging channel, which…
  - carries the event stream generated by the monitoring logic (i.e. the components OR a 3rd-party framework)
  - is accessible for the CEP platform



Notable implementations

- ESPER
- Drools Fusion (JBoss)
- IBM InfoSphere Streams (formerly: System S)
- Oracle CEP
- Microsoft StreamInsight
- StreamBase
- Lots of academic frameworks

Typical objection #1

- Q: „I'm a coder guy. Can't I just implement the event processing logic in my source code?"
- A: Yes, you can, but:
  - you will have to implement a custom processing algebra
  - as your event processing component evolves, first you will encounter the benefits of using *business rules* over if-else structures, then you will employ business roles with timing extensions for event processing

this is exactly how JBoss Drools Fusion works

Typical objection #2

- Q: „I'm all about databases. Can't I just put my events into a DB and write some queries and triggers to handle the event stream?"
- A: Yes, you can, but:
  - you will have to implement a custom processing algebra
  - in order to allow low-latency on-line processing, you will have to employ an in-memory DB and explore the benefits of the *active database* model

this is exactly how Esper works

Examples

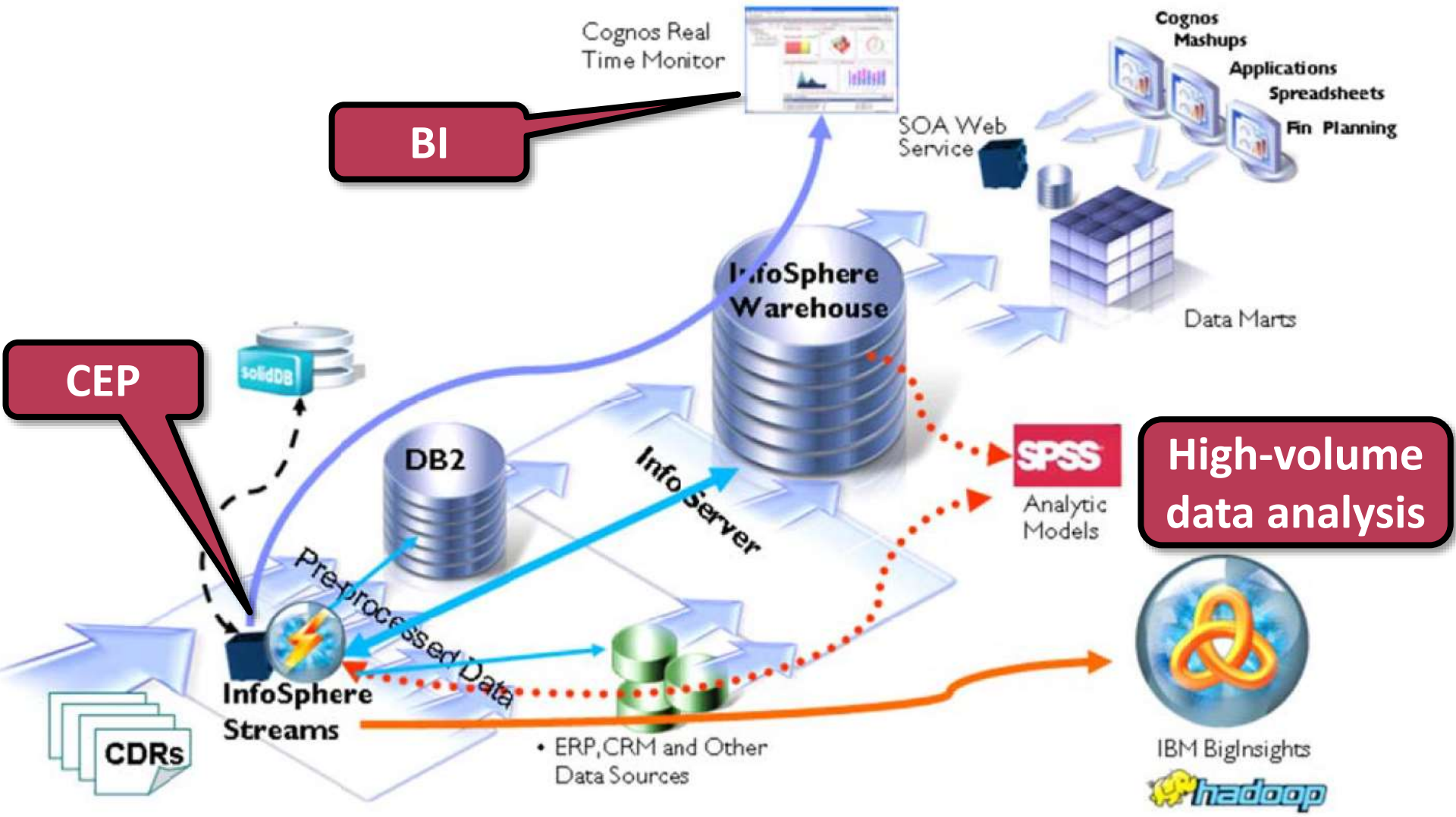
- Drools Fusion

```
rule „Rule1”
      when w:Withdrawal (amount>=200)
      then println(„Withdrawal over 200!");
end
```

- Esper

```
select * from
Withdrawal(amount>=200).WIN:LENGTH(5)
```

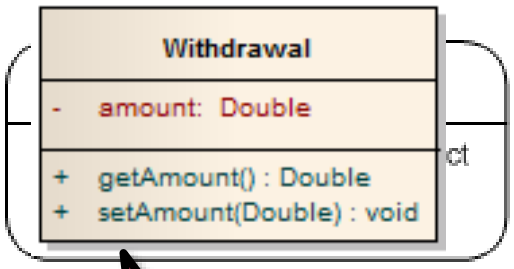
IBM Continuous Insight



ESPER

- Java-based implementation
- There is also an implementation for .NET (NEsper)
- Open-source
- Embeddable into your Java code
- You call it as a service, runtime
- Esper Processing Language (EPL)
  - SQL-like
  - Implements an expansive processing algebra
    - 400 pages of language definition, 100 pages of configuration

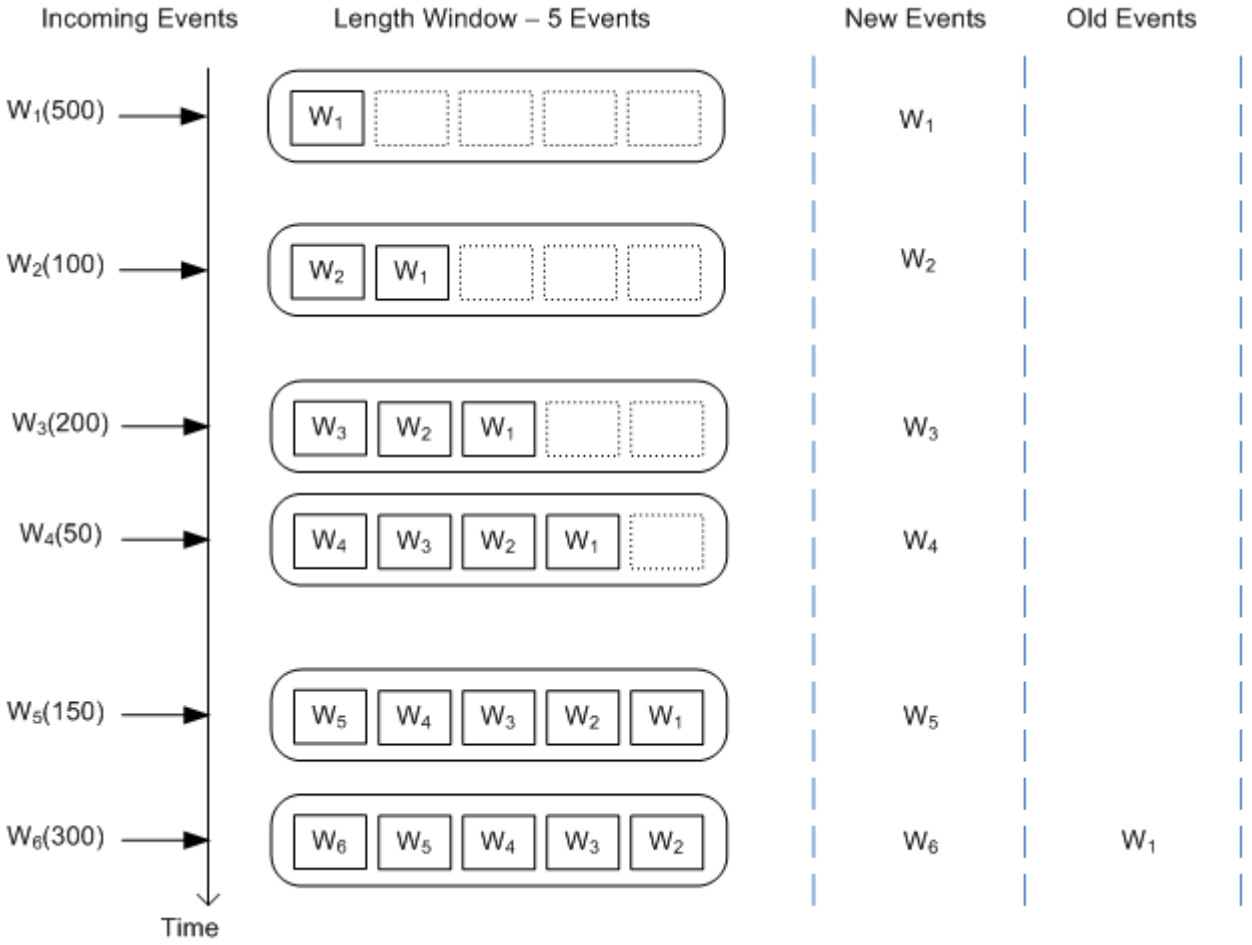
ESPER: Processing model



A proper Java bean

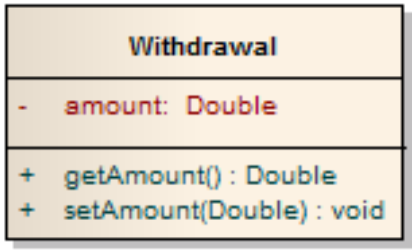
EPL expression

```
SELECT * FROM
Withdrawal.WIN:LENGTH(5)
```

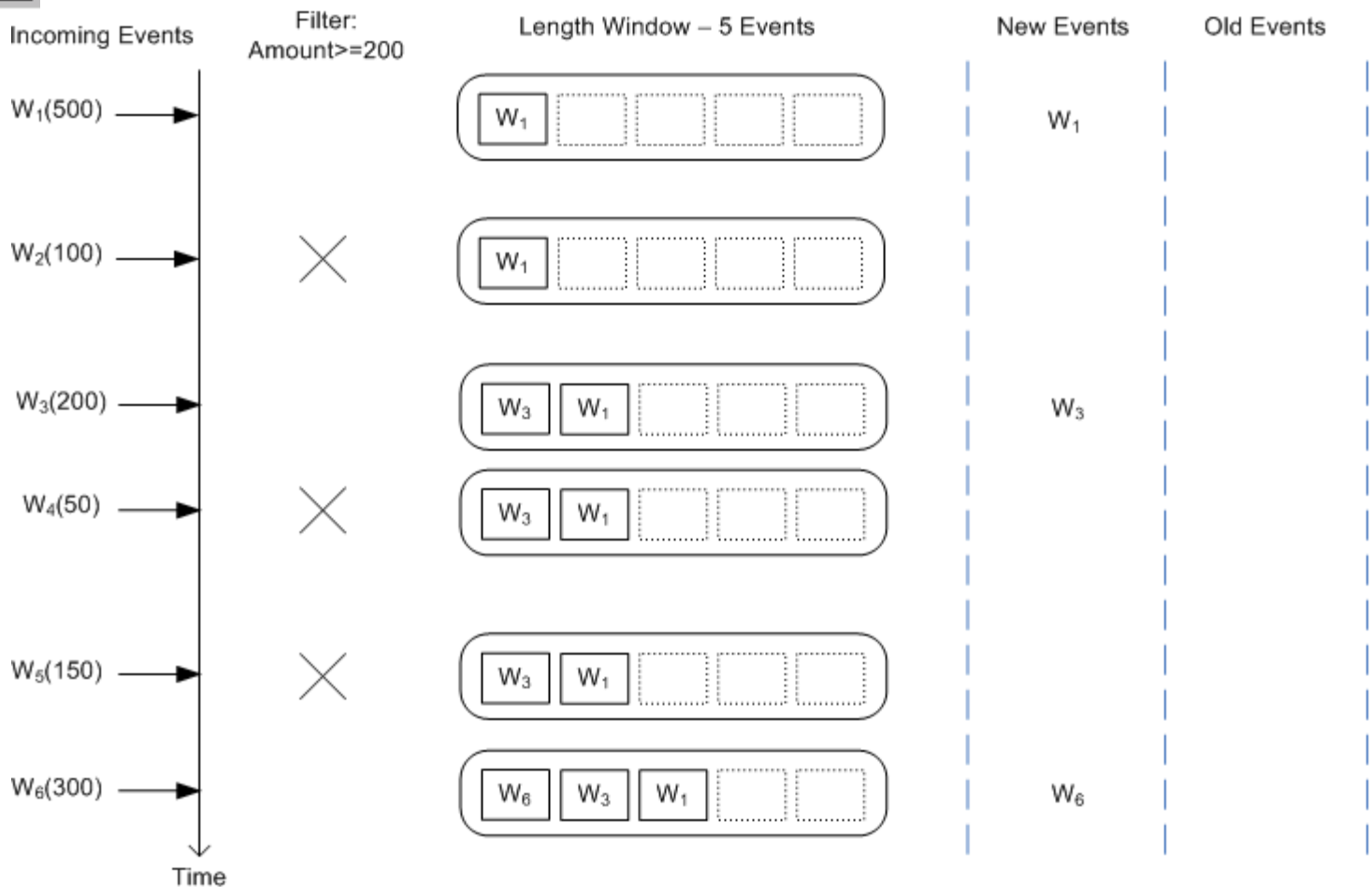


Executable action

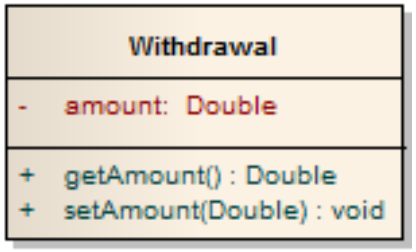
ESPER: Processing model



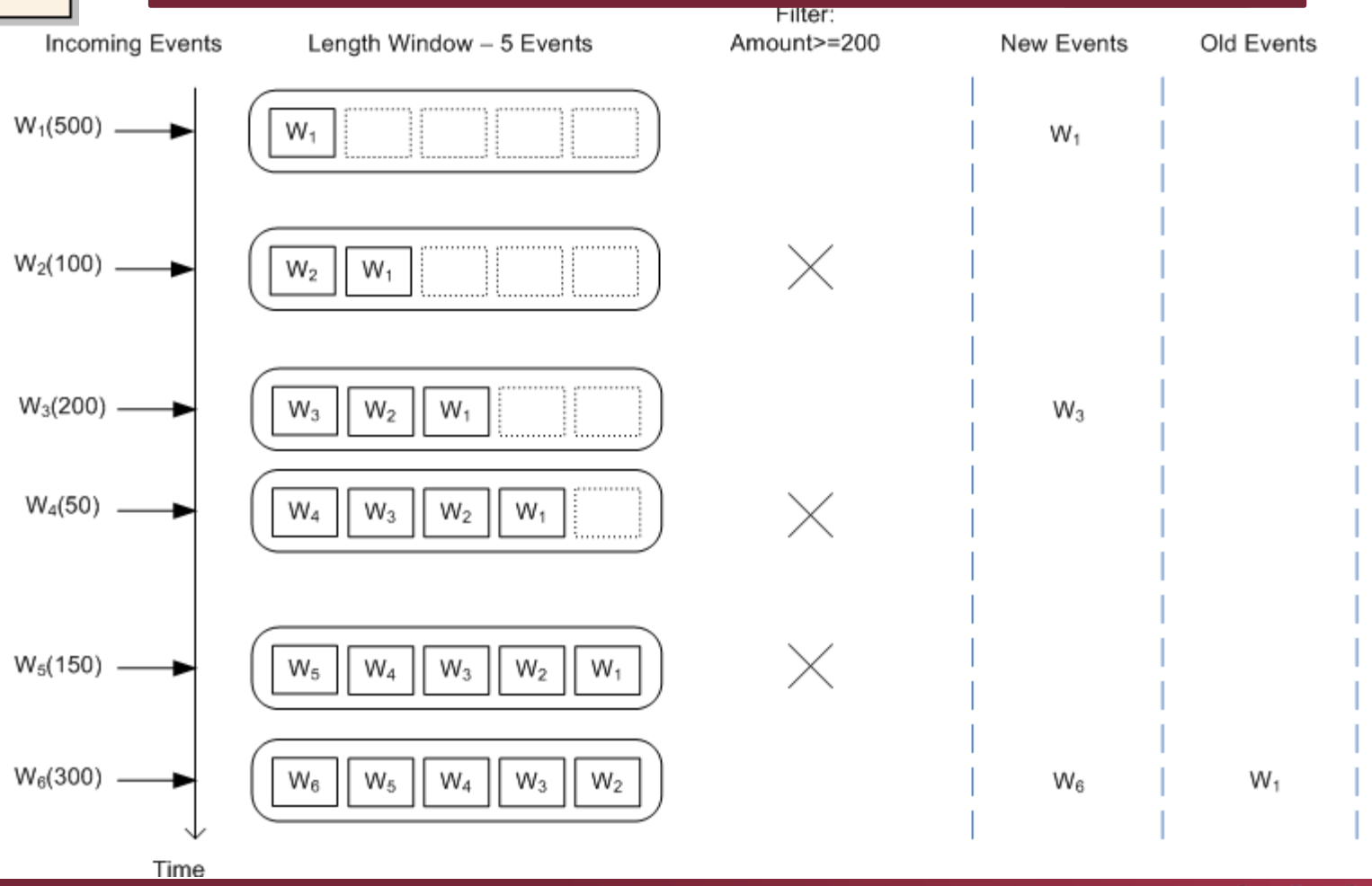

```
SELECT * FROM
Withdrawal(amount>=200).WIN:LENGTH(5)
```



ESPER: Processing model




```
SELECT * FROM
Withdrawal.WIN:LENGTH(5) WHERE amount>=200
```



Further Esper Pointers

The full documentation:
http://esper.codehaus.org/esper/documentation/documentation.html

Clauses, subqueries, patterns:
http://esper.codehaus.org/esper-4.7.0/doc/reference/en-US/html/epl_clauses.html

Complex patterns: http://esper.codehaus.org/esper-4.7.0/doc/reference/en-US/html/event_patterns.html

EPL operators in depth: http://esper.codehaus.org/esper-4.7.0/doc/reference/en-US/html/epl-operator.html

Design patterns:
http://esper.codehaus.org/tutorials/solution_patterns/solution_patterns.html (see section „EPL Questions”)

Steps for CEP @ESPER

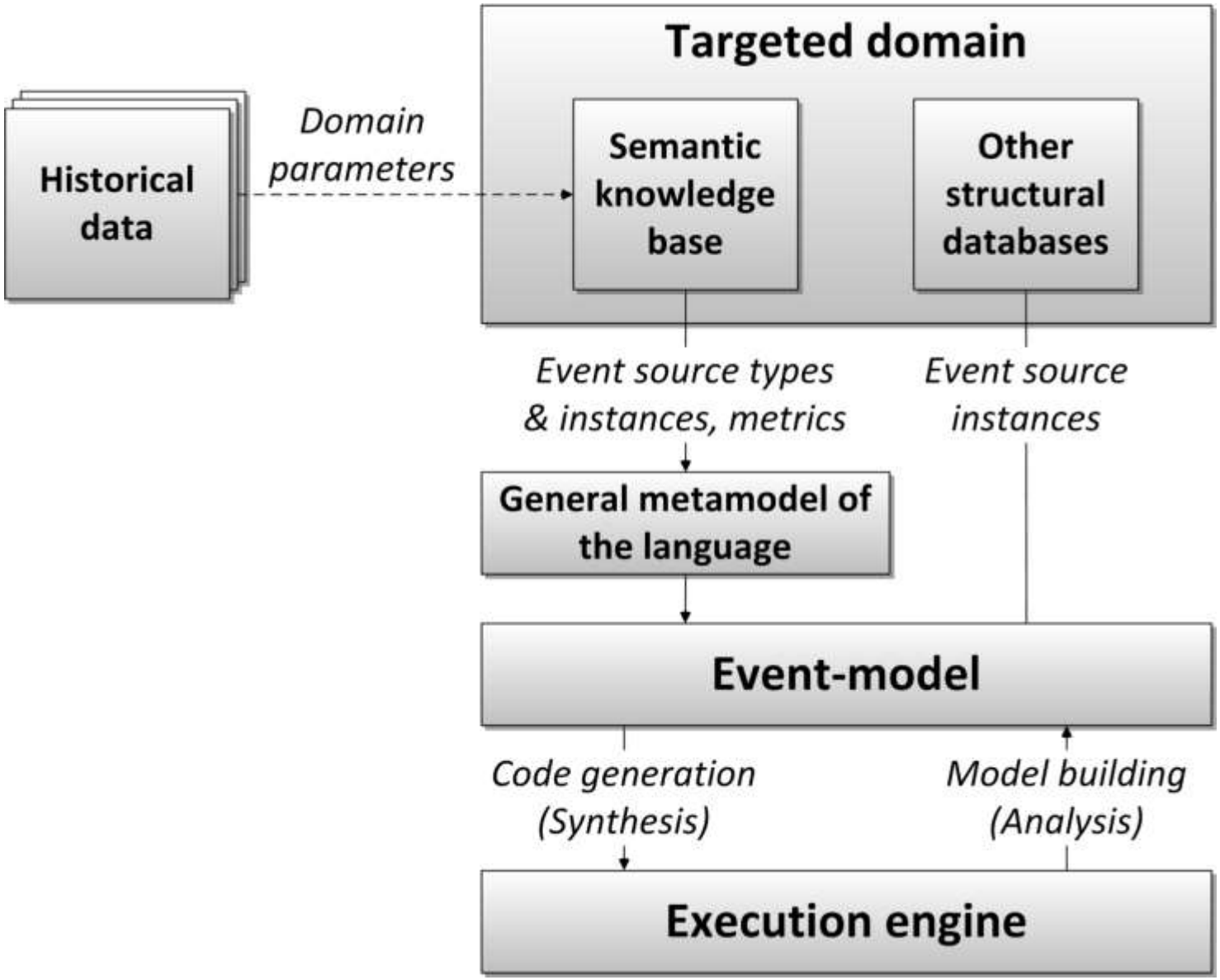
- Install the distribution
  - Download and unzip ☺
- In your Java project, import esper.jar
- Define Filters and Listeners
- Get an instance of the engine service provider
- Configuration
- Register Filters and Listeners
- Connect to the engine
- Send events

ESPER
DEMO

Emerging problems

- No modeling support
- Inconvenient coding
- Hard to maintain the codebase
- No validation for the defined patterns
  - Possible interference, overlaps, correlations, etc.
- Lack of traceability

The CEPWorkbench

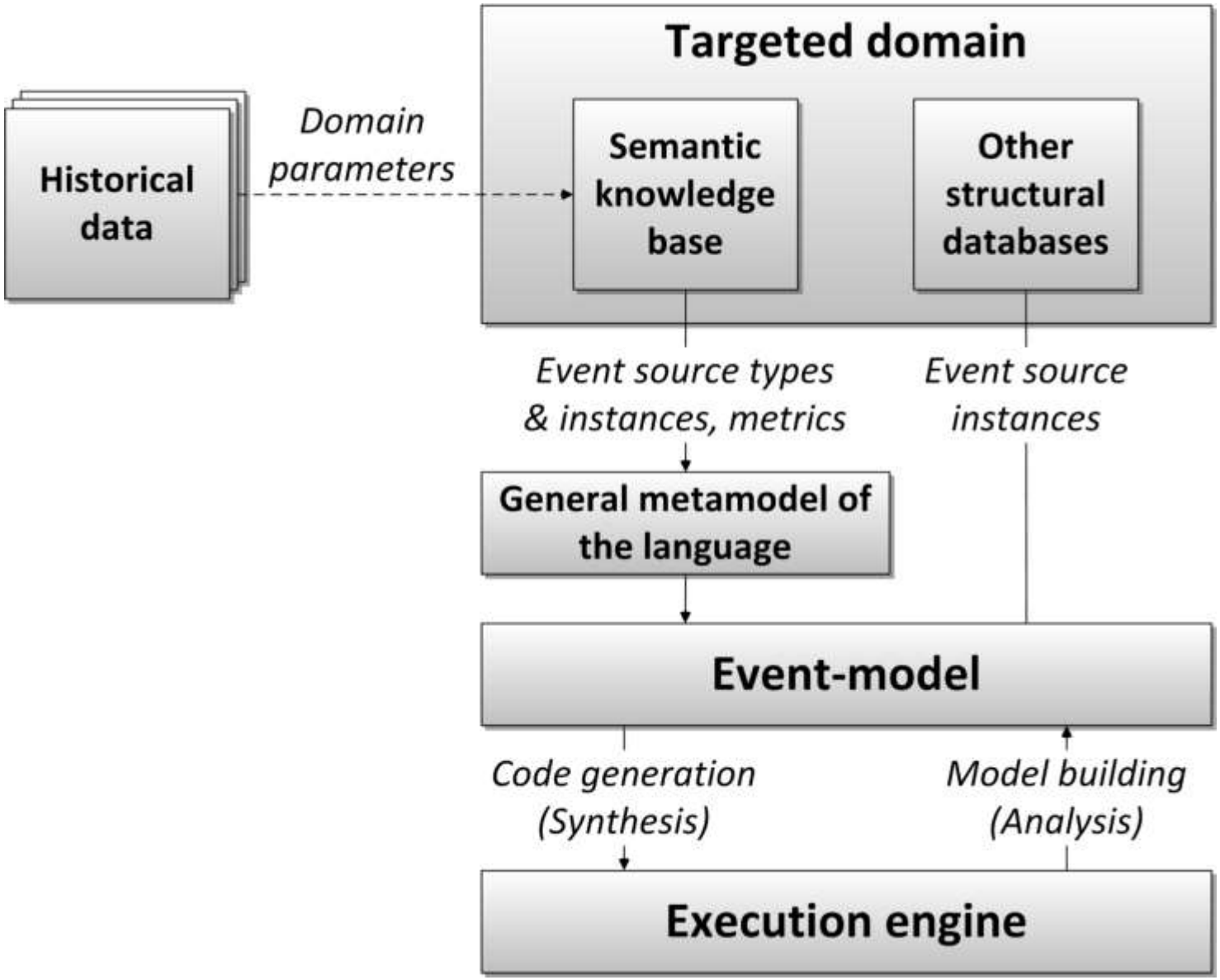


The CEPWorkbench

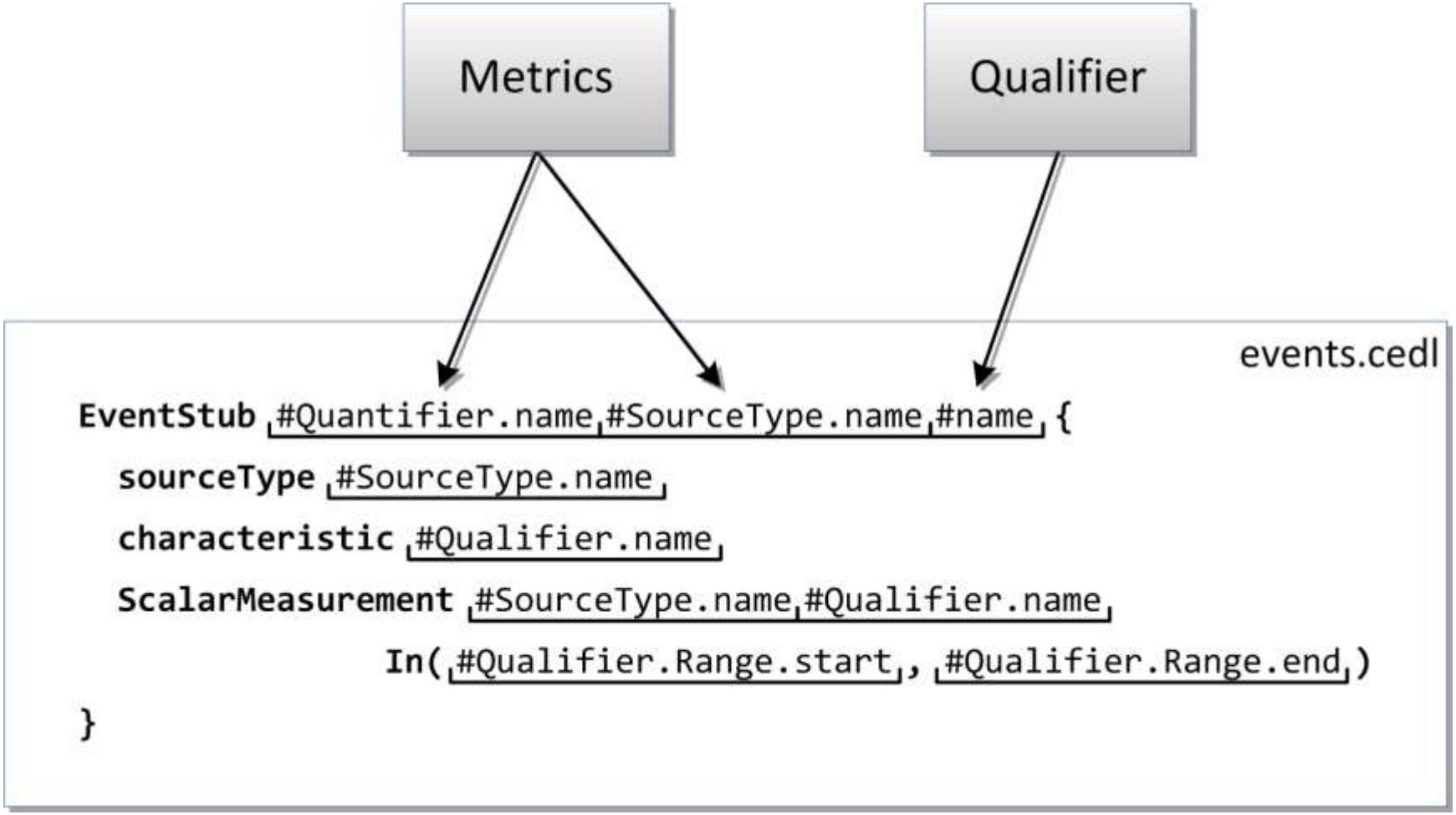
```
Event CPULoadCritical {
        source Server1
        PercentageMeasurement CPULoad Minimum 90
}
Event BackupProblem {
        source Server1
        ScalarMeasurement AvailableBackups LessThan 2
}


 ComplexEvent CriticalServer{
        CONCURRENT_T(CPULoadCritical BackupProblem; T: Minimum 30)
        action {
                Action of Type sendWarning
        }
 }
```

The CEPWorkbench



The CEPWorkbench



CEPWorkbench
DEMO

Future directions

- Models@run.time
- Enormous amount of data is produced day by day
  - The last 50 years: processing data at rest
  - The next 50 years: processing information in motion

  (Mark Palmer)
- *"The world is changing fast. Big will not beat small anymore. It will be the fast beating the slow."*
  (Rupert Murdoch)