JUDCon

JBoss Users & Developers Conference

Boston:2011

DEMISTIFYING COMPLEX EVENT PROCESSING

Edson Tirelli
Drools Fusion Lead, Red Hat
May 2nd, 2011

Agenda

- Brief introduction on CEP and Terminology
- Drools Fusion: Complex Event Processing extensions
  - Event Declaration and Semantics
  - Event Cloud, Streams and the Session Clock
  - Temporal Reasoning
  - Sliding Window Support
  - Streams Support
  - Memory Management
- Questions & Answers

Terminology: Event

"An event is an observable occurrence."

"An event in the Unified Modeling Language is a notable occurrence at a particular point in time."

http://www.wikipedia.org

"Anything that happens, or is contemplated as happening."

"An object that represents, encodes or records an event, generally for the purpose of computer processing"

http://complexevents.com

Terminology: Event

For the scope of this presentation:

"An event is a **significant change of state** at a particular **point in time**"

Terminology: Complex Event

"**Complex Event**, is an abstraction of other events called its members."

- Examples:
  - **The 1929 stock market crash** – an abstraction denoting many thousands of member events, including individual stock trades)
  - **The 2004 Indonesian Tsunami** – an abstraction of many natural events
  - **A completed stock purchase** -an abstraction of the events in a transaction to purchase the stock
  - **A successful on-line shopping cart checkout** – an abstraction of shopping cart events on an on-line website

- *Source: http://complexevents.com*

Terminology: CEP

"**Complex Event Processing**, or CEP, is primarily an event processing concept that deals with the task of processing multiple events with the goal of **identifying the meaningful events** within the event cloud.

CEP employs techniques such as **detection** of complex patterns of many events, event **correlation** and **abstraction**, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes."

-- wikipedia

Terminology: CEP

- Examples:
  - Emergency Response Systems
  - Credit Card Fraud Detection
  - Logistics Real-Time Awareness solution
  - Neonatal ICU: infant vital signs monitoring

Terminology: CEP vs ESP

Complex Event Processing, or CEP, and **Event Stream Processing**, or ESP, are two technologies that were born separate, but converged.

- An *oversimplification*: In their origins...
  - Event Stream Processing focused on the ability to process high volume **streams** of events.
  - Complex Event Processing focused on defining, detecting and processing the **relationships** among events.

Terminology: CEP and ESP

For the scope of this presentation:

"CEP is used as a common term meaning both CEP and ESP."

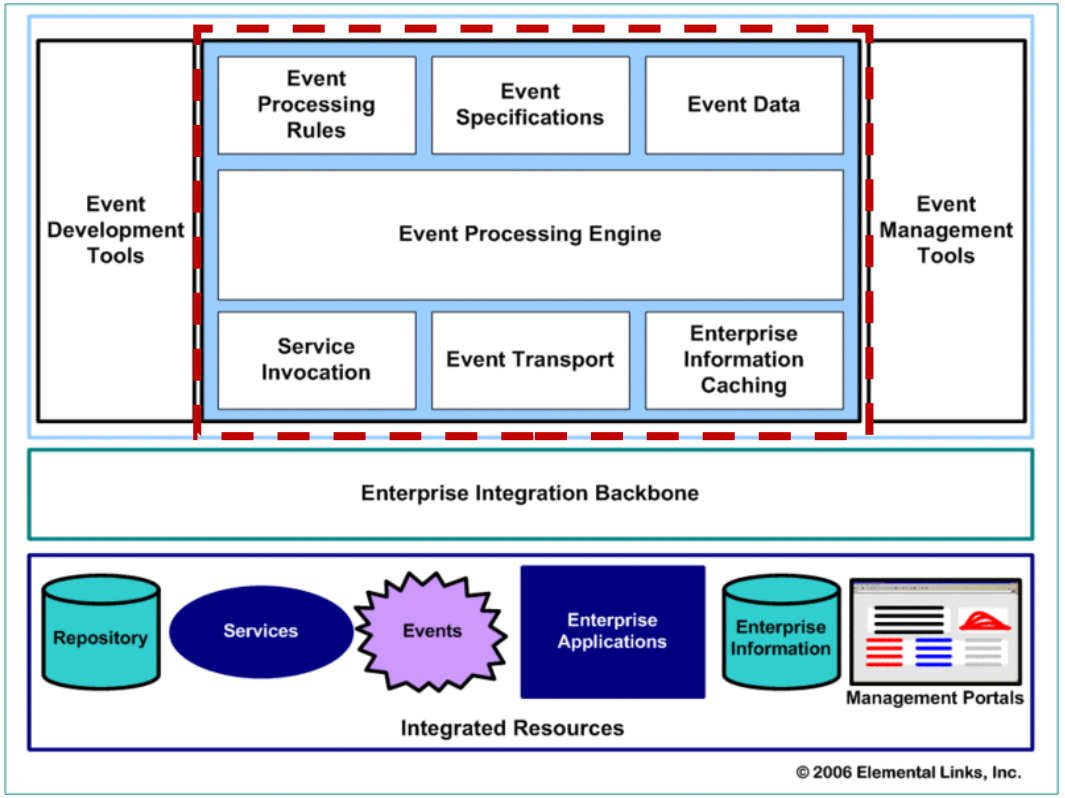
Terminology: EDA

"**Event Driven Architecture (EDA)** is a software architecture pattern promoting the production, detection, consumption of, and reaction to events. An event can be defined as "a significant change in state"[1]. For example, when a consumer purchases a car, the car's state changes from "for sale" to "sold". A car dealer's system architecture may treat this state change as an event to be produced, published, detected and consumed by various applications within the architecture."

http://en.wikipedia.org/wiki/Event_Driven_Architecture

EDA vs CEP

CEP is a **component** of the EDA

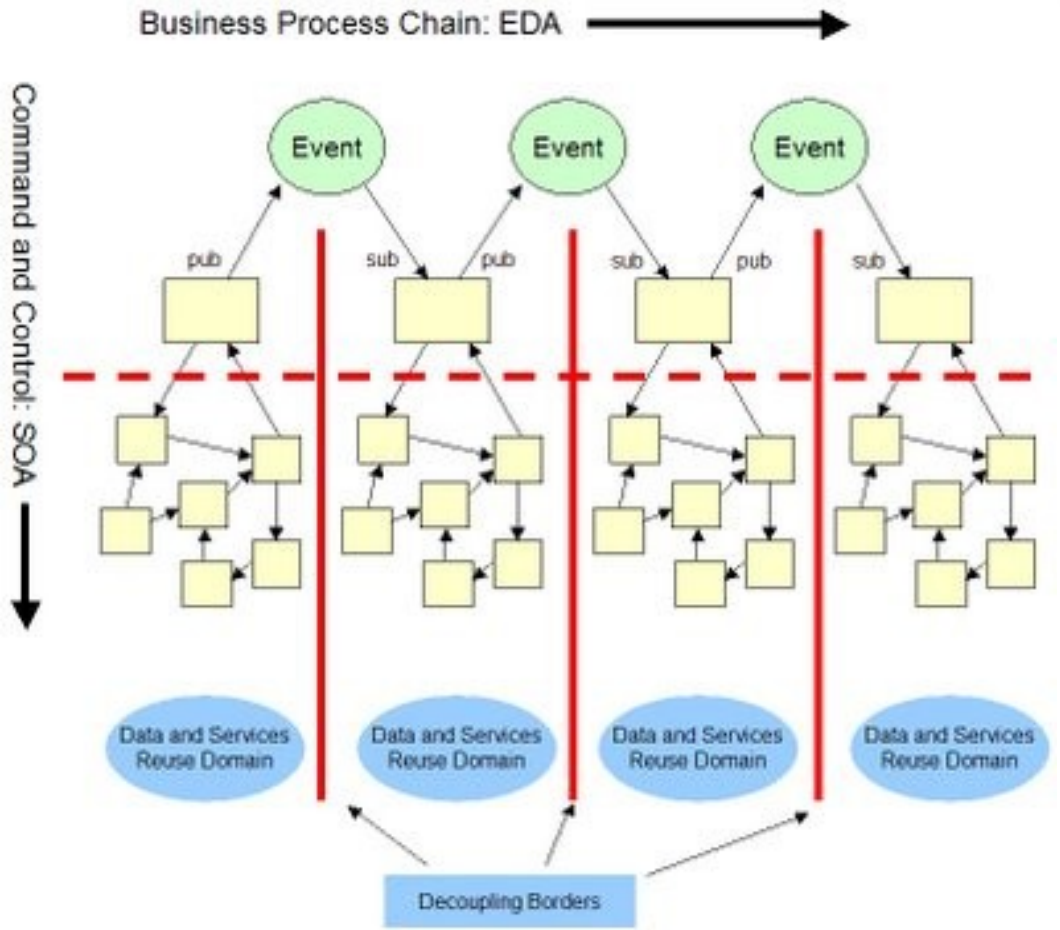


Source: http://elementallinks.typepad.com/.shared/image.html?/photos/uncategorized/simple_event_flow.gif

EDA vs SOA

- EDA is **not** SOA 2.0
- Complementary architectures
- Metaphor
  - In our body:
    - SOA is used to build our muscles and organs
    - EDA is used to build our sensory system

EDA vs SOA



Source: http://soa-eda.blogspot.com/2006/11/how-eda-extends-soa-and-why-it-is.html

Complex Event Processing

- **A few characteristics of common CEP scenarios:**
  - Huge volume of events, but only a few of real interest
  - Usually events are immutable
  - Usually queries/rules have to run in reactive mode
  - Strong temporal relationships between events
  - Individual events are usually not important
  - The composition and aggregation of events is important

Drools Vision

Guvnor

Expert

Fusion

Flow

“A common platform to model and govern the business logic of the enterprise.”

JBoss Enterprise BRMS v5.2

Drools Fusion: Enables…

- **Event Detection:**
  - From an event cloud or set of streams, select all the meaningful events, and only them.
- **[Temporal] Event Correlation:**
  - Ability to correlate events and facts declaring both temporal and non-temporal constraints between them.
  - Ability to reason over event aggregation
- **Event Abstraction:**
  - Ability to compose complex events from atomic events AND reason over them

Drools Fusion

- **Features:**
  - Event Semantics as First Class Citizens
  - Allow Detection, Correlation and Composition
  - Temporal Constraints
  - Session Clock
  - Stream Processing
  - Sliding Windows
  - CEP volumes (scalability)
  - (Re)Active Rules
  - Data Loaders for Input

Demo

- **Twitter Stream CEP Demo:**
  - Listen to the Twitter Stream API
    - Twitter4J API
    - Listens to a random sample of tweets
  - Detects patterns and reacts
    - Drools Fusion
  - Simple one process (multi-thread) demo
    - Focus on specific features

Event Declaration and Semantics

```
// declaring existing class
import some.package.VoiceCall
declare VoiceCall
  @role( event )
  @timestamp( calltime )
  @duration( duration )
end

// generating an event class
declare StockTick
  @role( event )

  symbol : String
  price : double
end
```

Event semantics:

- Point-in-time and Interval

An event is a fact with a few special characteristics:

- Usually immutable, but not enforced
- Strong temporal relationships
- Lifecycle may be managed
- Allow use of sliding windows

“All events are facts, but not all facts are events.”

Temporal Reasoning

- Semantics for:
  - **time:** discrete
  - **events:** point-in-time and interval
- Ability to express temporal relationships:
  - Allen's 13 temporal operators

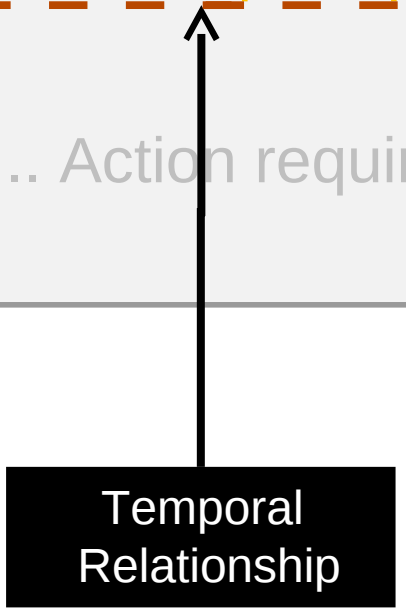
- **James F. Allen** defined the 13 possible temporal relations between two events.
- **Eiko Yoneki** and **Jean Bacon** defined a unified semantics for event correlation over time and space.

Temporal Relationships

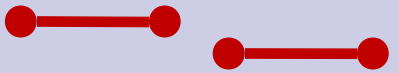
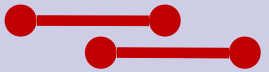
```
rule “Shipment not picked up in time”
when
    Shipment( $pickupTime : scheduledPickupTime )
    not ShipmentPickup( this before $pickupTime )
then
    // shipment not picked up... action required.
end
```

Temporal Relationships

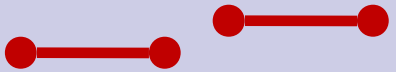
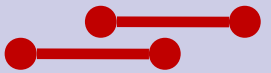
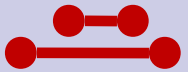
```
rule “Shipment not picked up in time”
when
    Shipment( $pickupTime : scheduledPickupTime )
    not ShipmentPickup( this before $pickupTime )
then
    // shipment not picked up... Action required.
end
```



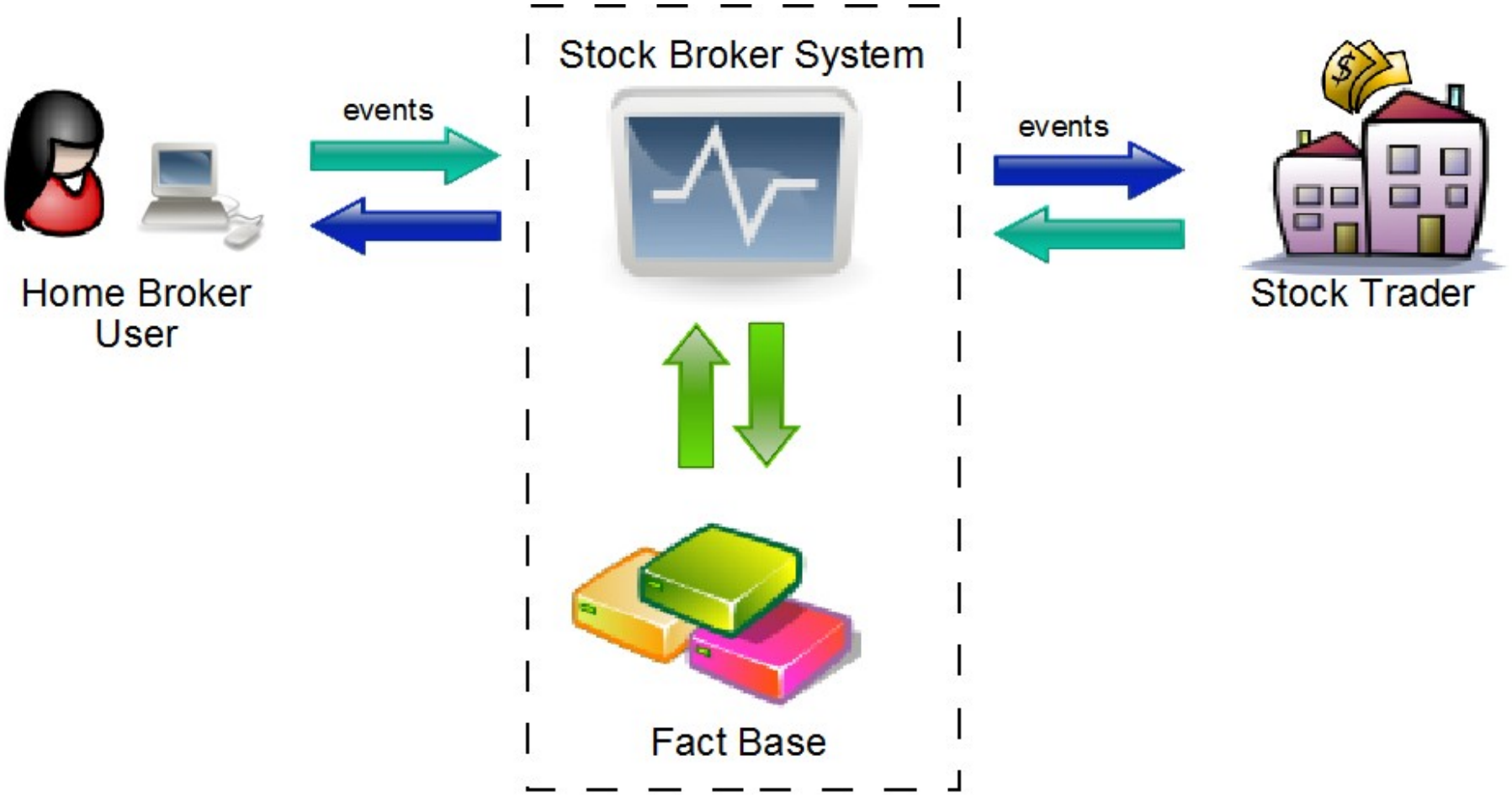
Allen's 13 Temporal Operators

	Point-Point	Point-Interval	Interval-Interval
A before B (A / B)			
A meets B (A / B)			
A overlaps B (A / B)			
A finishes B (A / B)			
A includes B (A / B)			
A starts B (A / B)			
A coincides B (A / B)			

Allen's 13 Temporal Operators

	Point-Point	Point-Interval	Interval-Interval
A after B (A / B)			
A metBy B (A / B)			
A overlapedBy B (A / B)			
A finishedBy B (A / B)			
A during B (A / B)			
A finishes B (A / B)			

Streams : Simple Example Scenario



Stream Support (entry-points)

- **A scoping abstraction for stream support**
  - Rule compiler gather all entry-point declarations and expose them through the session API
  - Engine manages all the scoping and synchronization behind the scenes.

```
rule “Stock Trade Correlation”
when
    $c : Customer( type == “VIP” )
    BuyOrderEvent( customer == $c, $id : id ) from entry-point “Home Broker Stream”
    BuyAckEvent( sourceEvent == $id ) from entry-point “Stock Trader Stream”
then
    // take some action
end
```

Cloud Mode, Stream Mode, Session Clock

CLOUD	STREAM
• No notion of "flow of time": the engine sees all facts without regard to time • No attached Session Clock • No requirements on event ordering • No automatic event lifecycle management • No sliding window support	• Notion of "flow of time": concept of "now" • Session Clock has an active role synchronizing the reasoning • Event Streams must be ordered • Automatic event lifecycle management • Sliding window support • Automatic rule delaying on absence of facts

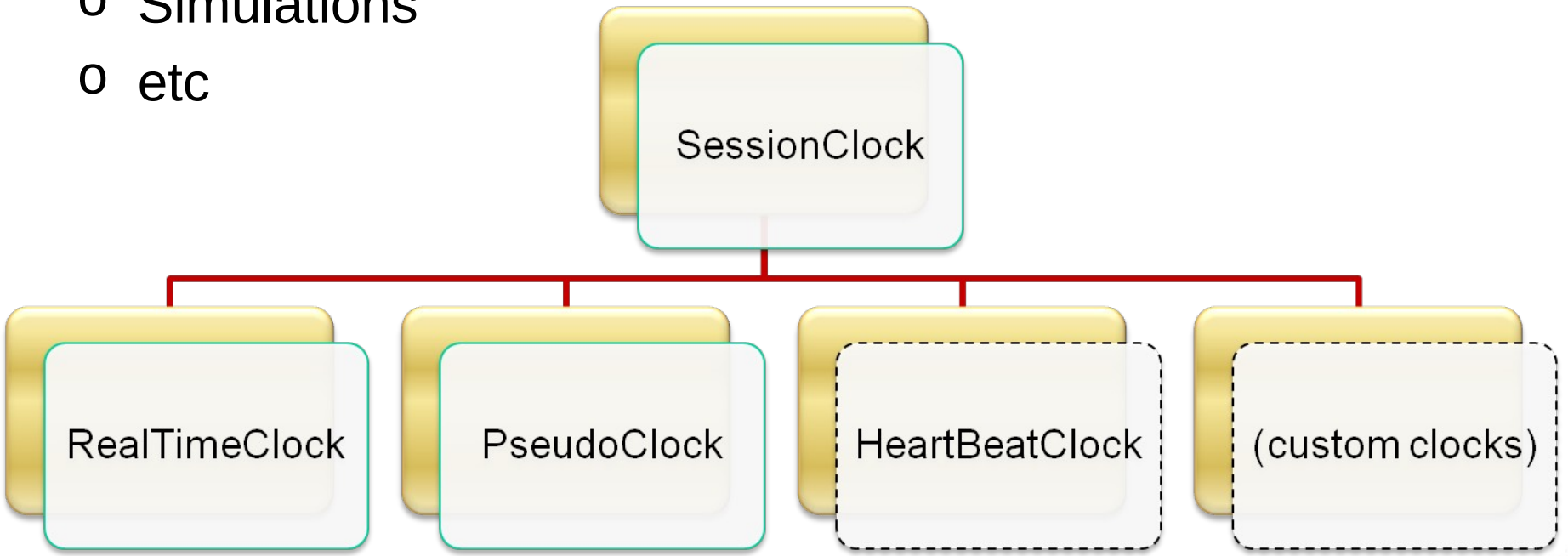
Reference Clock

- Reference clock defines the flow of time

- Named **Session Clock**
  - is assigned to each session created
- Synchronizes time sensitive operations
  - duration rules
  - event streams
  - process timers
  - sliding windows

Session Clock

- Uses the strategy pattern and multiple implementations:
  - Real-time operation
  - Tests
  - Simulations
  - etc



Session Clock

- Selecting the session clock:
  - API:

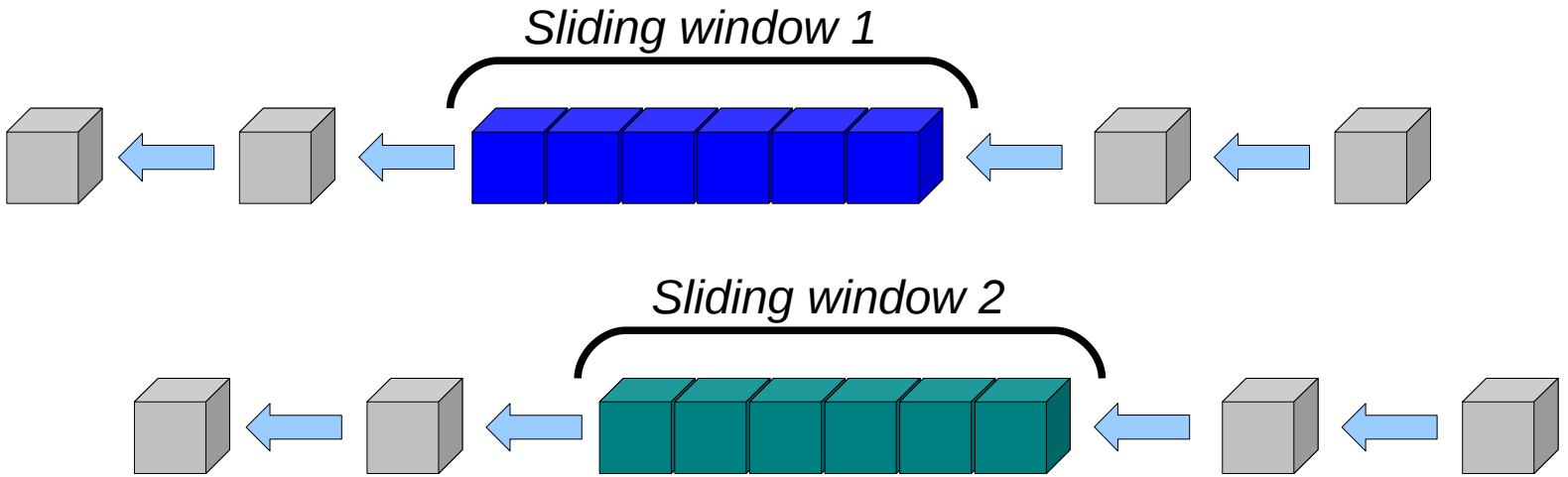
```
KnowledgeSessionConfiguration conf = ...
conf.setOption( ClockTypeOption.get( “realtime” ) );
```

  - System Property or Configuration File:

```
drools.clockType = pseudo
```

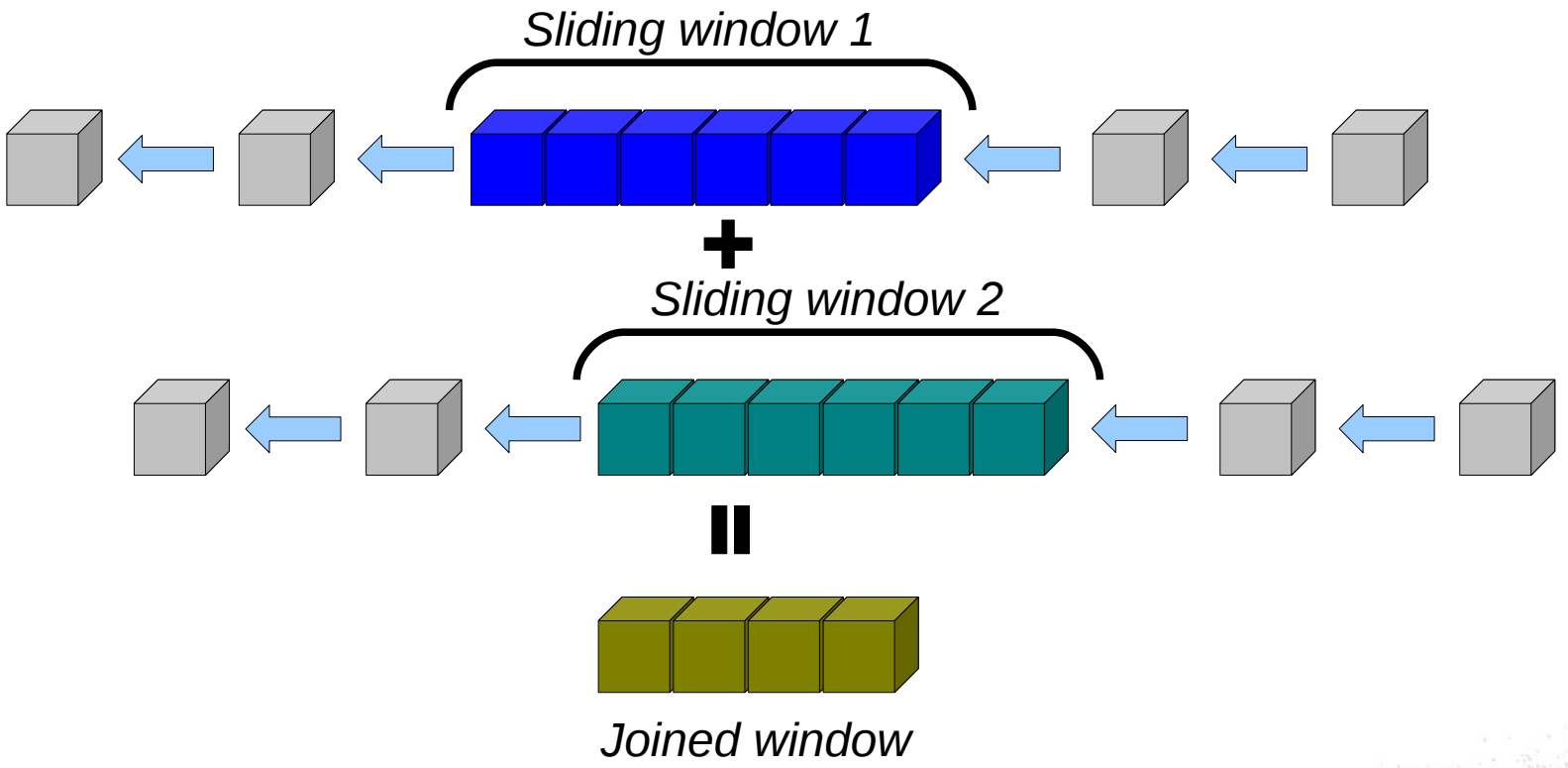
Sliding Window Support

- Allows reasoning over a moving window of “interest”
  - Time
  - Length



Sliding Window Support

- Allows reasoning over a moving window of “interest”
  - Time
  - Length



Delaying Rules

- Negative patterns may require rule firings to be delayed.

```
rule “Order timeout”
when
    $bse : BuyShares ( $id : id )
    not BuySharesAck( id == $id, this after[0s,30s] $bse )
then
    // Buy order was not acknowledged. Cancel operation
    // by timeout.
end
```

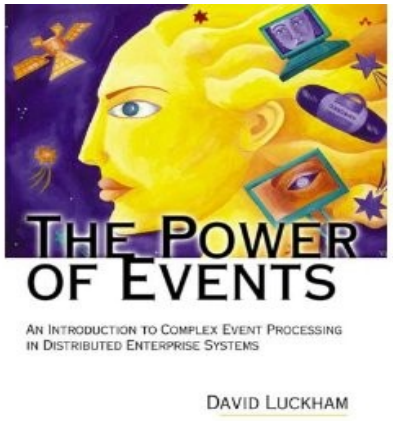
Temporal Dimension

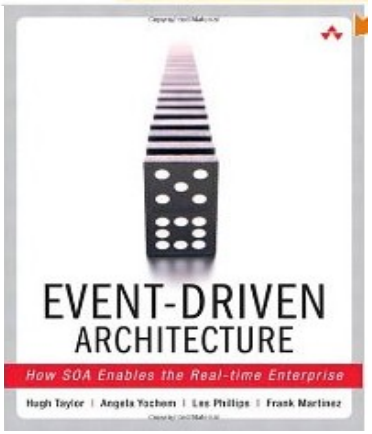
- Requires the support to the temporal dimension
  - A rule/query might match in a given point in time, and not match in the subsequent point in time

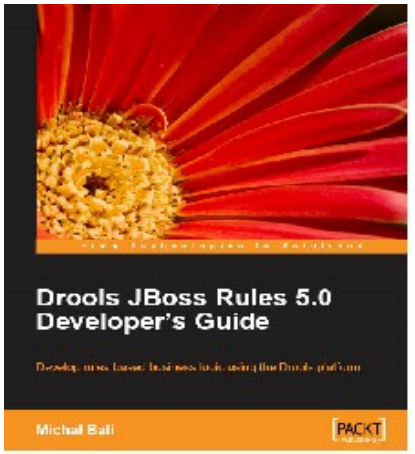
- That is the single most difficult requirement to support in a way that the engine:
  - stays deterministic
  - stays a high-performance engine
- Achieved mostly by compile time optimizations that enable:
  - constraint tightening
  - match space narrowing
  - memory management

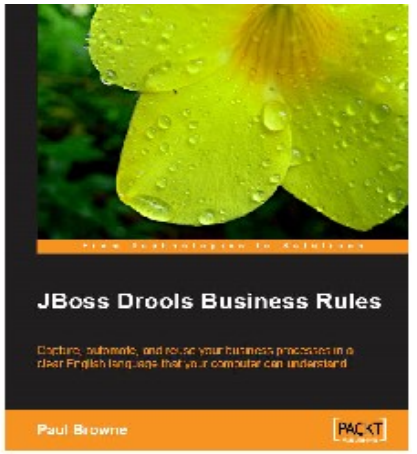
Q&A

- **Drools project site:**
  - http://www.drools.org (http://www.jboss.org/drools/)
- **Documentation:**
  - http://www.jboss.org/drools/documentation.html









Edson Tirelli – *etirelli@redhat.com*

JUDCon

JBoss Users & Developers Conference

Boston:2011