Temporal Reasoning:
A requirement for CEP

Edson Tirelli
etirelli@redhat.com
Lead CEP Engineer
JBoss, a Division of Red Hat

Adam Mollenkopf
adam.mollenkopf@fedex.com
Strategic Technologist
FedEx Custom Critical

Agenda

- **Presentation**
- Brief introduction on CEP
- CEP Applied at FedEx Custom Critical
- Drools Vision
- Temporal requirements for CEP
- Temporal extensions for Complex Event Processing
  - Session Clock
  - Temporal relationships
  - Sliding Window Support
  - Side-effect: Memory Management made possible
- Questions & Answers

Presentation

- **The spirit:**
  - "No fluff, just stuff!"
  - "Open source, open minds!"
- **The references:**
  - "Not a complete bibliography, just references to a few sources for the curious."
- **IMPORTANT DISCLAIMER**
  - CEP use cases add a whole set of functional and non-functional requirements
    - Too many to cover in a single session.
  - Temporal Reasoning is just ONE of these requirements and the focus of this presentation.

Complex Event Processing

“**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

Complex Event Processing

- **A few characteristics of common CEP scenarios:**
  - Huge amount 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

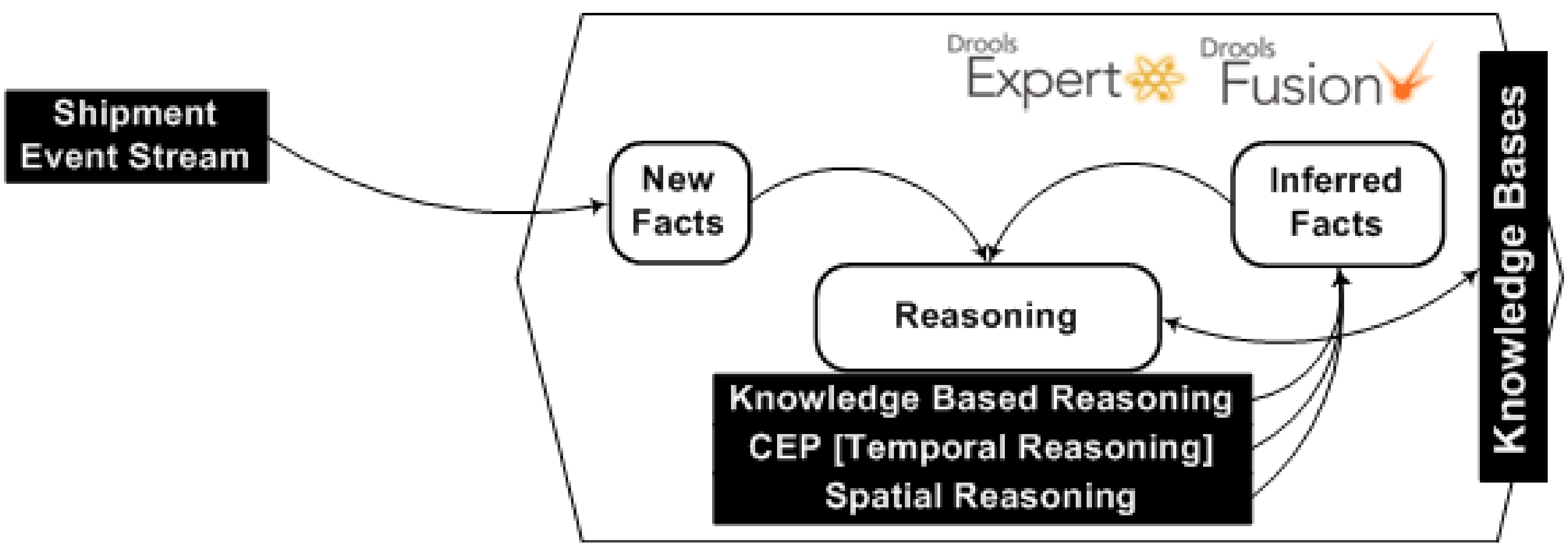
CEP Applied at FedEx Custom Critical

Business Problem Overview

- Time specific deliveries for critical freight
- Exclusive use non-stop door-to-door services
- Blended Surface and Air services to minimize cost and transit time
- Extra care in handling and specially equipped vehicles
  - Temperature Control, Secured Services, Hazardous Material, Constant Surveillance

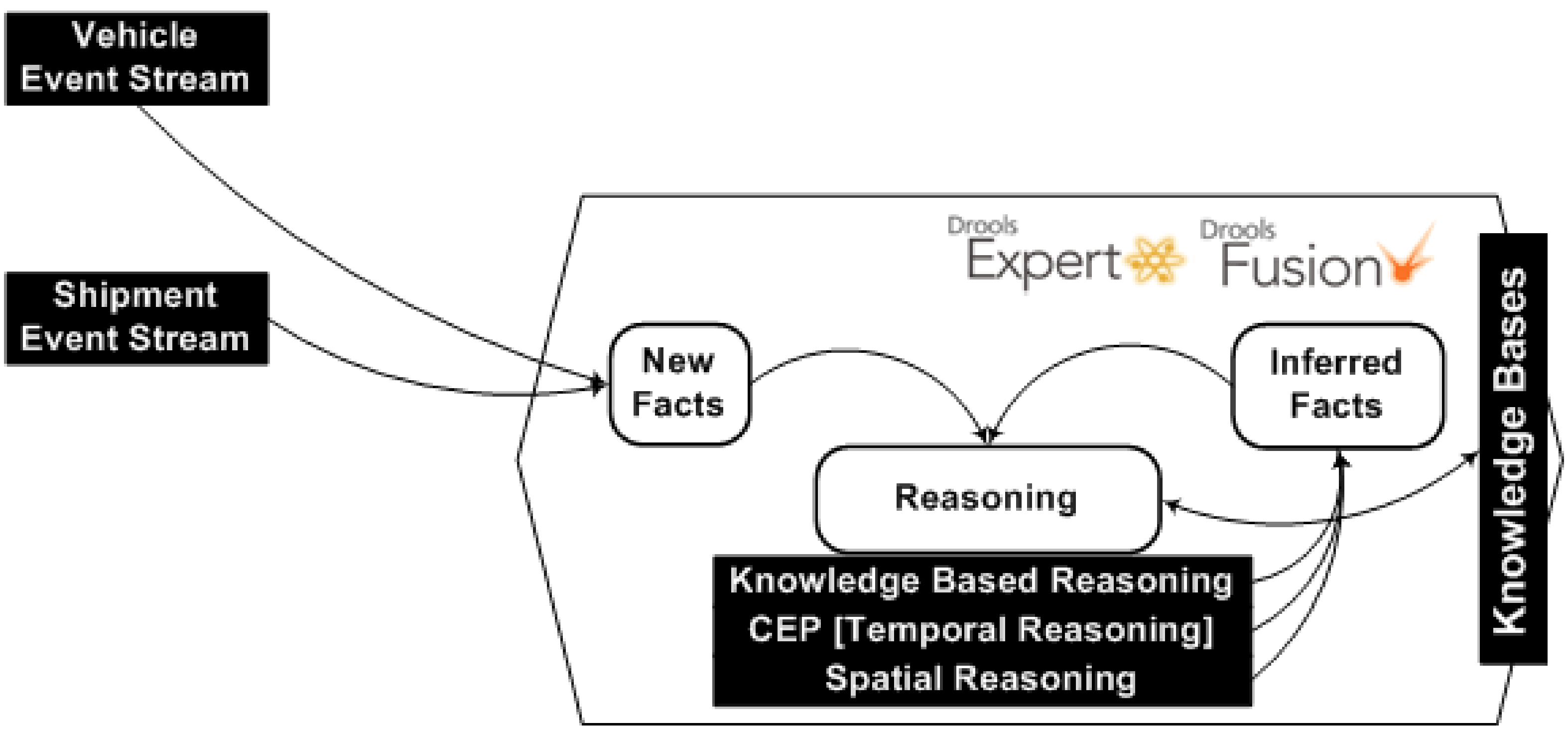
CEP Applied at FedEx Custom Critical

Business Problem – Types of Events



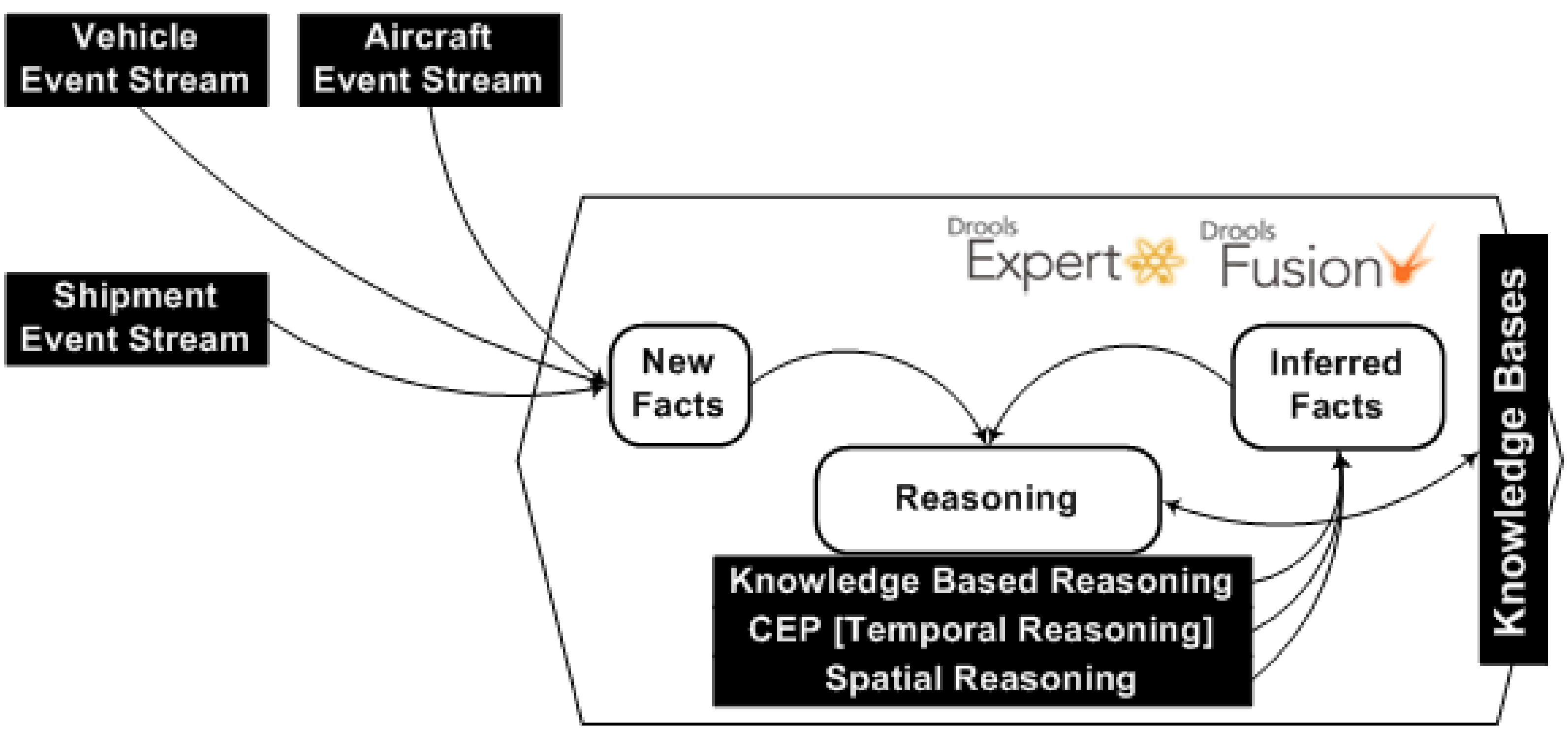
CEP Applied at FedEx Custom Critical

Business Problem – Types of Events



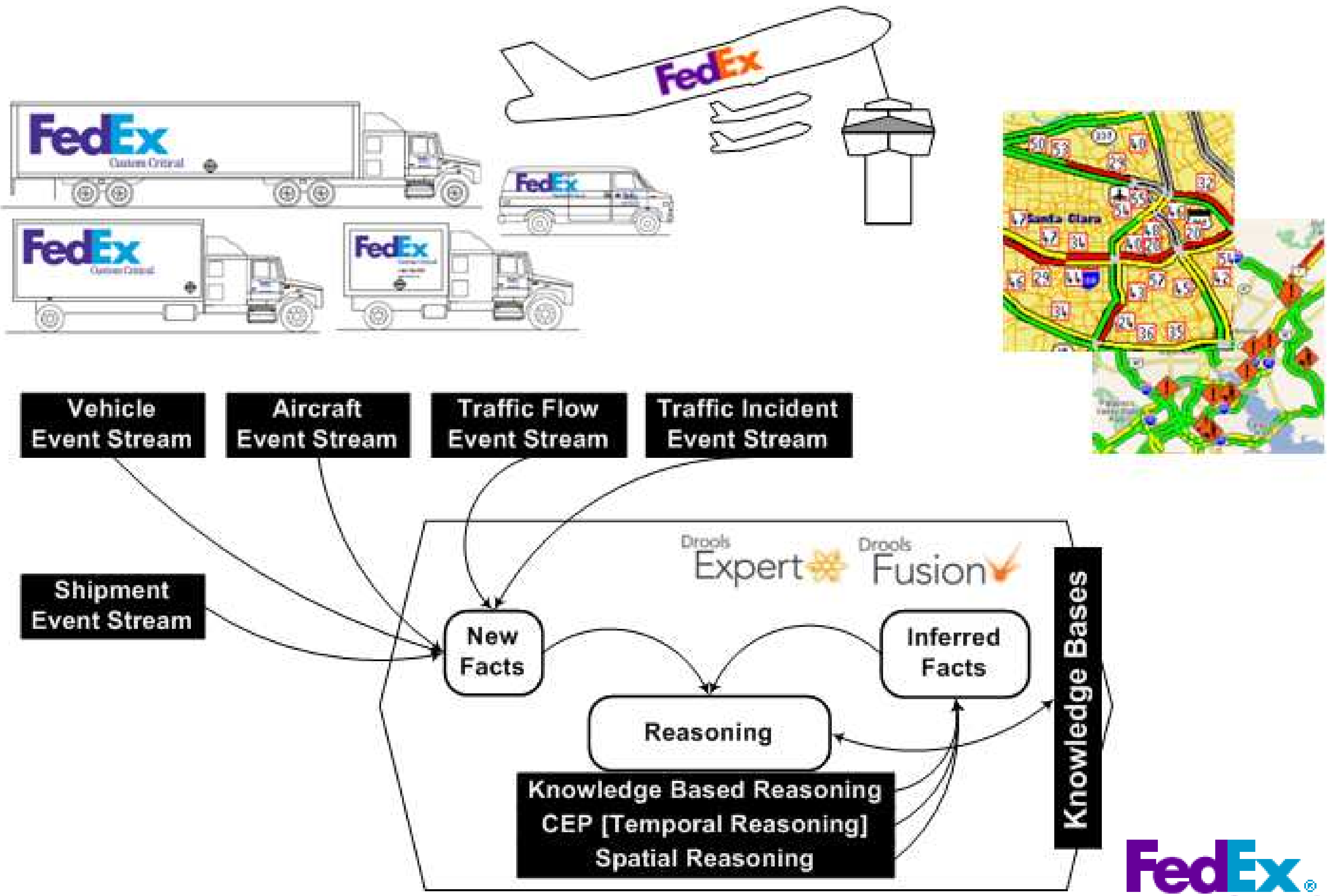
CEP Applied at FedEx Custom Critical

Business Problem – Types of Events



CEP Applied at FedEx Custom Critical

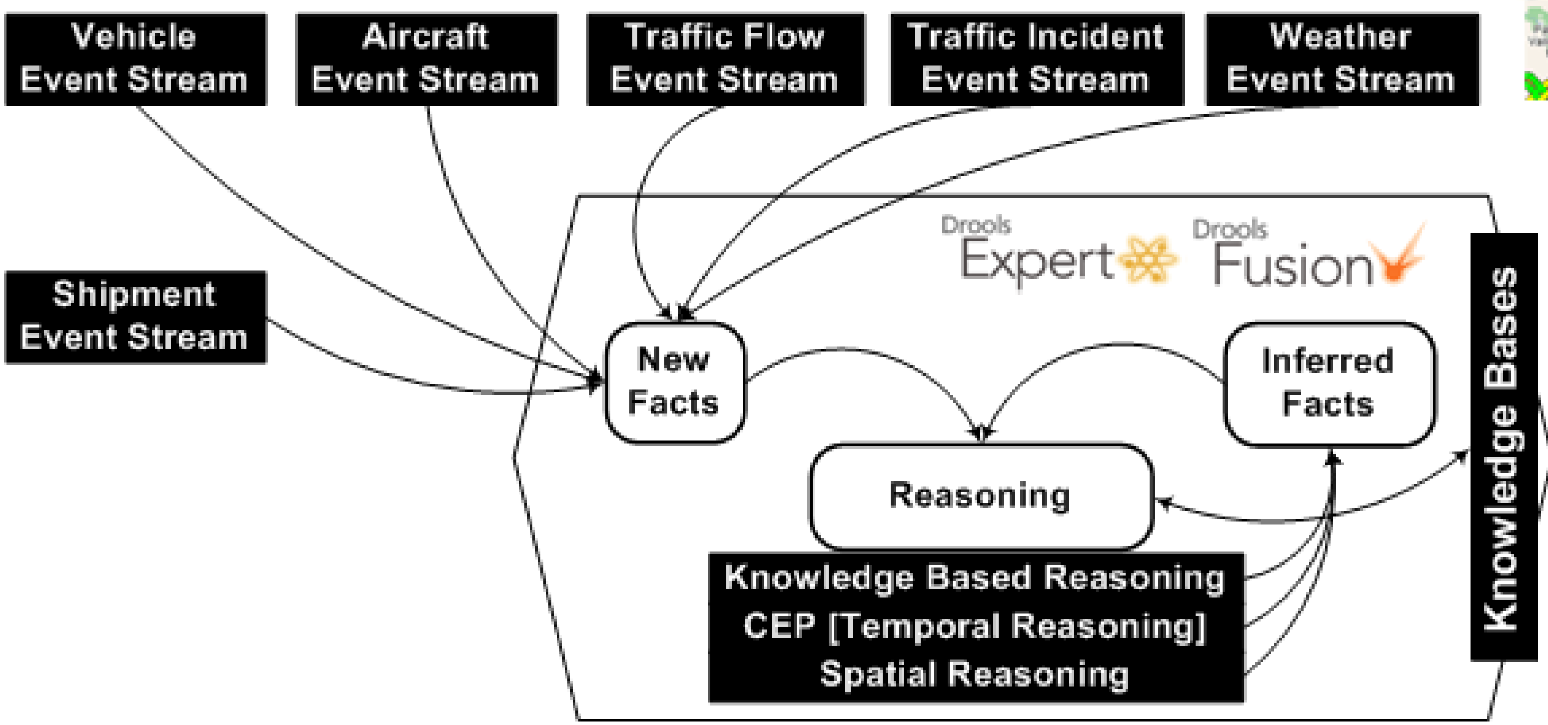
Business Problem – Types of Events



FedEx Custom Critical

CEP Applied at FedEx Custom Critical

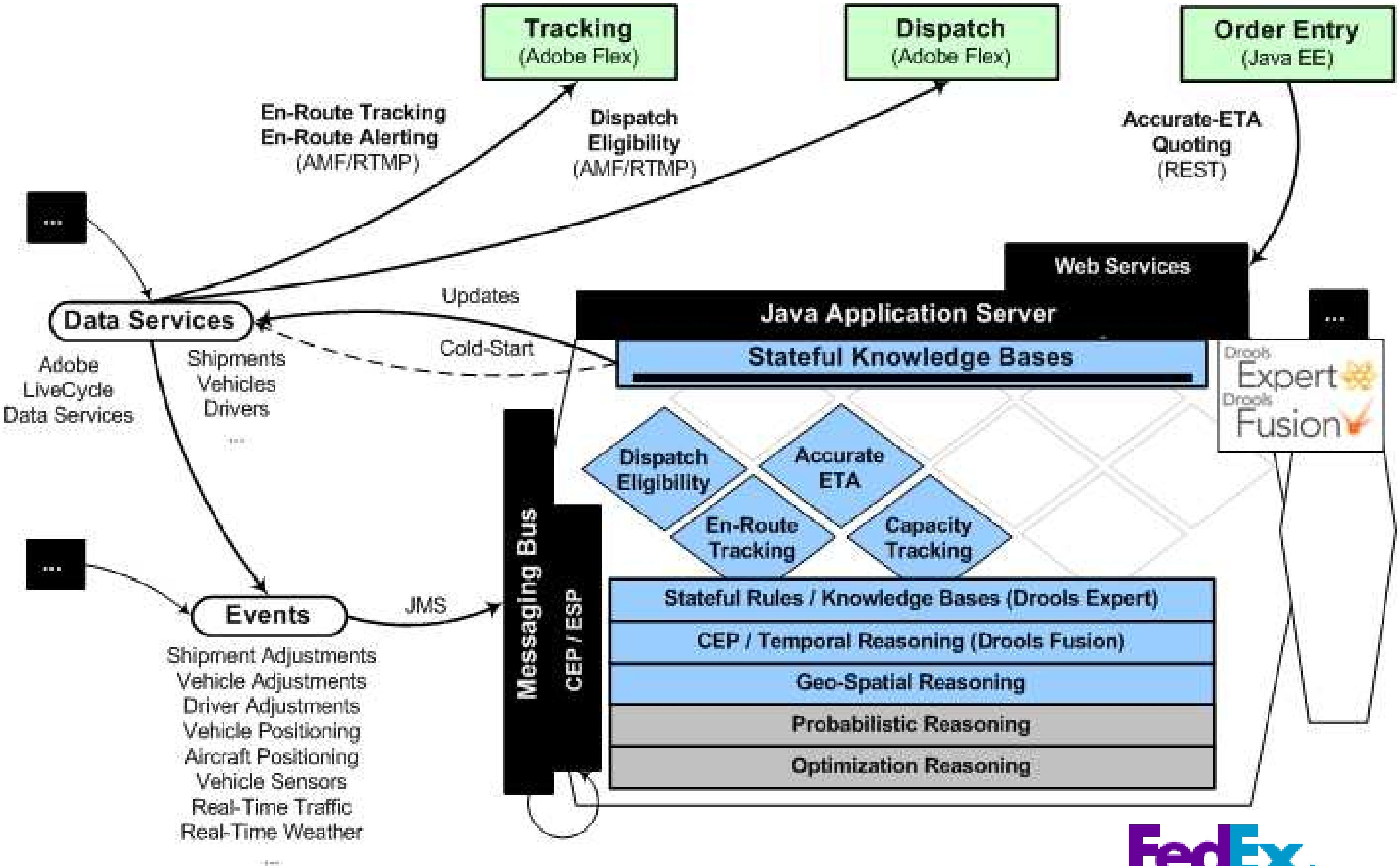
Business Problem – Types of Events



Demonstration

CEP Applied at FedEx Custom Critical

Stateful Rules Engine + CEP = Real-Time Intelligence



CEP Applied at FedEx Custom Critical

Anticipated Benefits

- At least 50% of Alerts can be reasoned automatically, promoting staff savings and improved Customer and Driver experiences.
- Risk Avoidance via pro-active monitoring
  - Reduction in insurance claims and shipment service failures
- Minimum 30% efficiency gains in shipment monitoring , saving at least 15% of Operations staff cost.

Question: what this has to do with anything to be presented in a conference about Business Rules technologies?

Drools Vision

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

Motivation

- Business Rules, Event Processing and Business Processes are all modelled declaratively.
- A business solution usually involves the interaction between these technologies.
- In short:
  - Technology overlap
  - Business overlap
- Several (good) products on the market:
  - Better either at CEP/ESP or Rules Processing or Business Processes
- The approach: attribute the same importance to the three complementary business modeling techniques

Drools Fusion

- **Goals:**
  - 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

Temporal Reasoning

1. Requires the definition of the semantics for:
   - **time:** discrete, dense or continuous
   - **events:** point-in-time or interval
2. Requires the ability to express temporal relationships
3. Requires the use of a reference clock
   - Implementation of time-flow
4. 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

Drools Fusion

- **Goals:**
  - 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

Temporal Reasoning Support Features (Temporal Constraints, Session Clock, Stream Processing, Sliding Windows)

Semantics, Temporal Relationships

1. Semantics for:
   - **time:** discrete
   - **events:** point-in-time and interval
2. 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
```

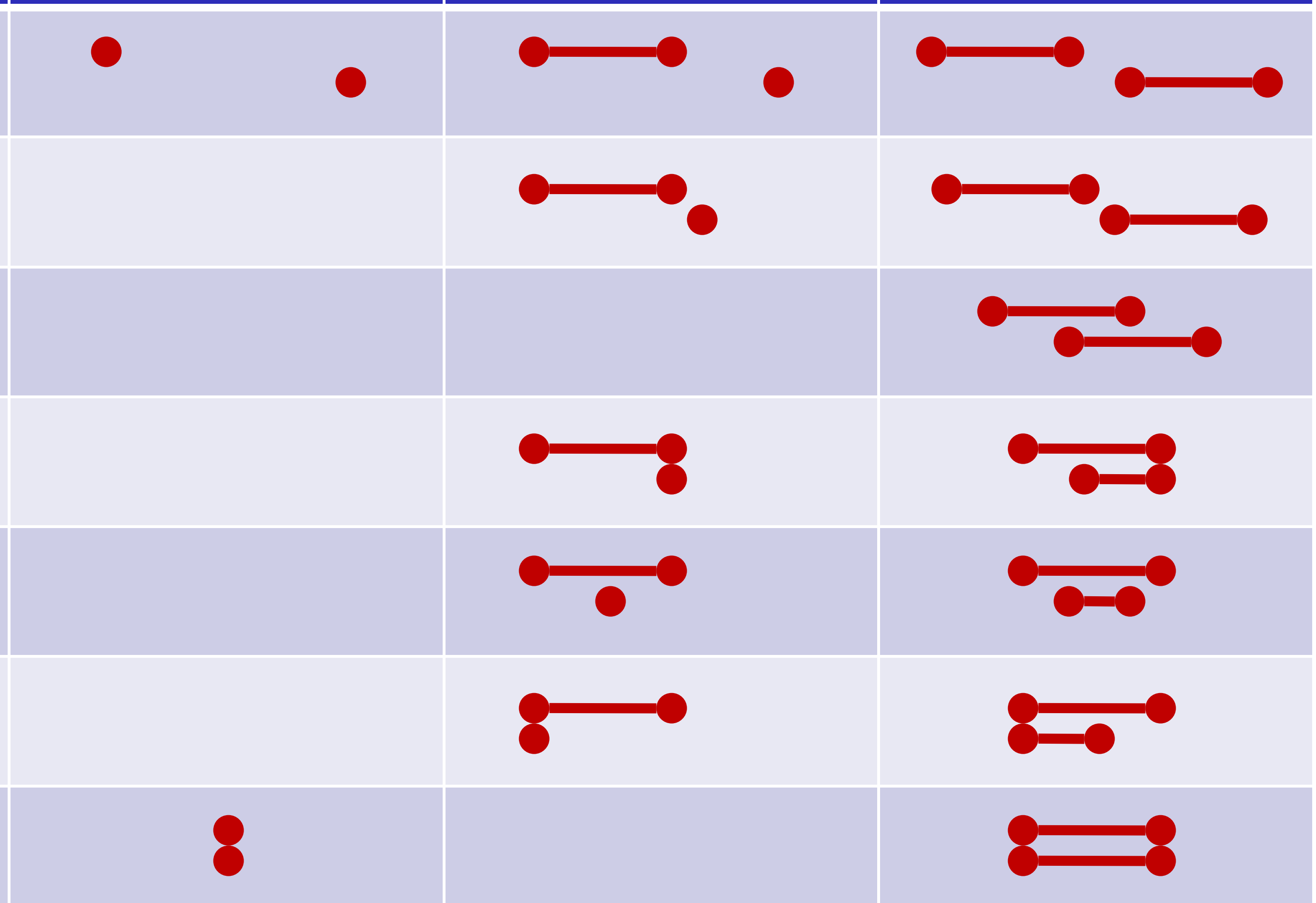
Temporal Relationships

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

Temporal Relationship

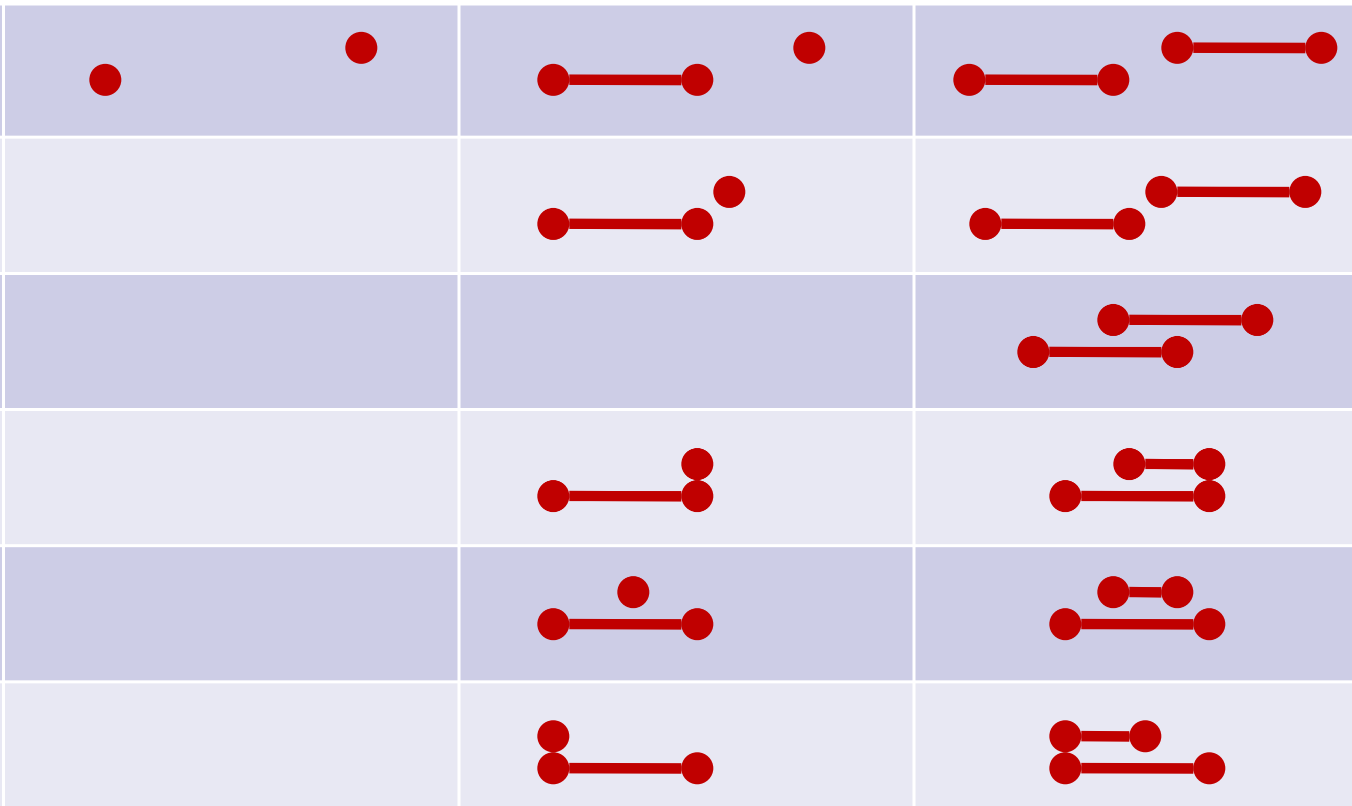
Allen's 13 Temporal Operators

	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 metBy B			
A overlapedBy B			
A finishedBy B			
A during B			
A finishes B			



Some references

- **Allen, J. F**. *An interval-based representation of temporal knowledge*. 1981.
- **Allen, J. F**. *Maintaining knowledge about temporal intervals*. 1983
- **Yoneki, Eiko and Bacon, Jean**. *Unified Semantics for Event Correlation Over Time and Space in Hybrid Network Environments*. 2005.
- **Bennett, Brandon and Galton, Antony P**. *A Unifying Semantics for Time and Events*. 2000.

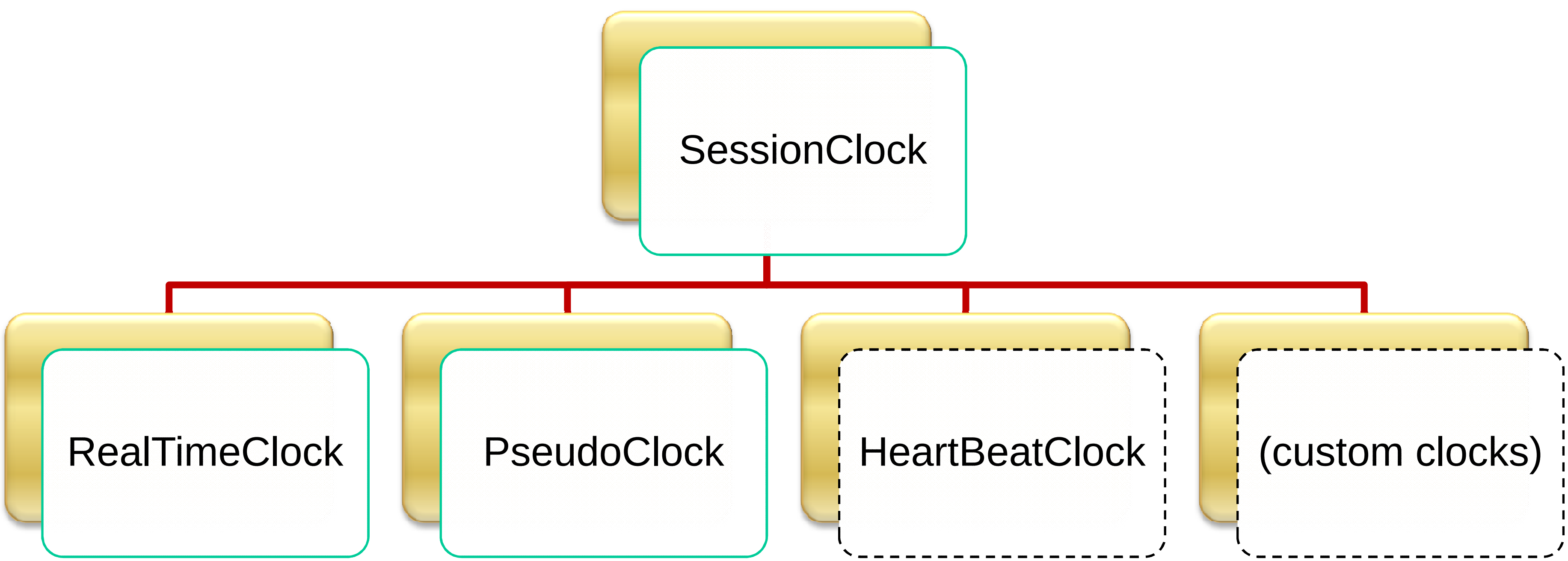
Reference Clock

3. Defines a reference clock
   - Implementation of time-flow

- 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

```
rule “Average Order Value over 12 hours”
when
    $c : Customer()
    $a : Number() from accumulate (
            BuyOrder( customer == $c, $p : price ) over window:time( 12h ),
            average( $p ) )
then
    // do something
end
```

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
```

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
```

Forces the rule to wait for 30 seconds before firing, because the acknowledgement may arrive at any time!

Some references

- **Ghanem, Hammad, Mokbel, Aref and Elmagarmid**. *Incremental Evaluation of Sliding-Window Queries over Data Streams*.

Temporal Dimension

4. 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

Temporal Dimension Support

- CEP scenarios are stateful by nature.
- Events usually are only interesting during a short period of time.
- Hard for applications to know when events are not necessary anymore
  - Temporal constraints and sliding windows describe such “window of interest”

Simple Example Rule

```
rule “Bag was not lost”
when
    $c : BagEvent(  ) from entry-point “check-in”
    $l : BagEvent( this == $c.bagId, this after[0,5m] $c )
                           from entry-point “pre-load”
then
    // bag was not lost
end
```

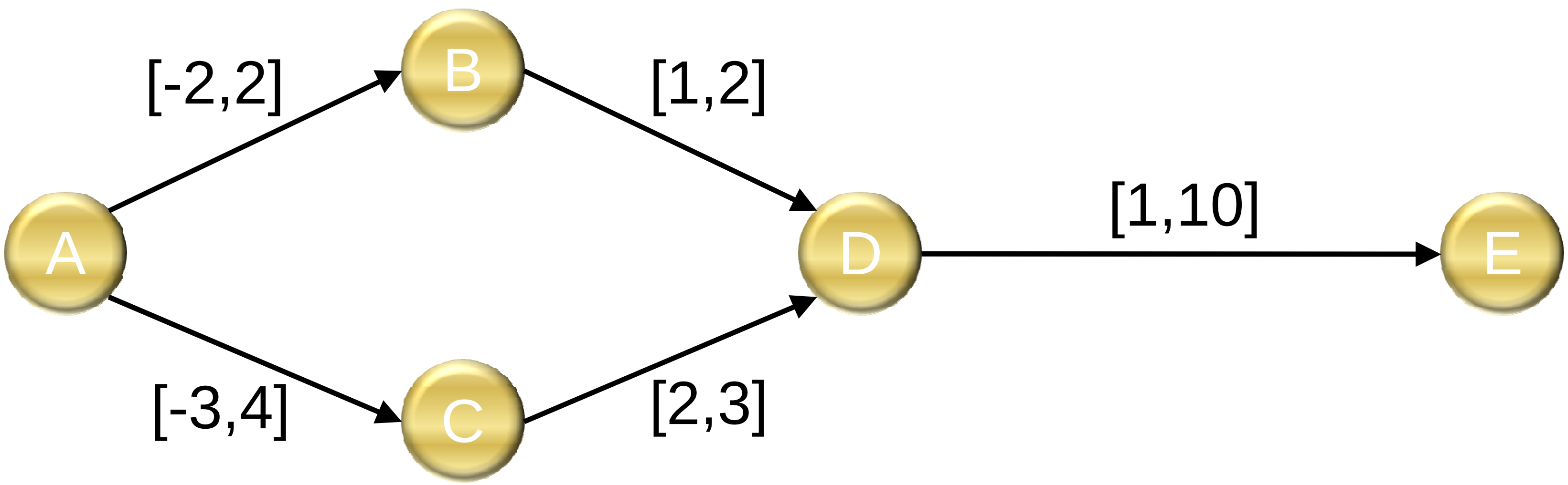
Abstract Example Rule

```
rule "reasoning on events over time"
when
    $a : A( )
    $b : B( this after[-2,2] $a )
    $c : C( this after[-3,4] $a )
    $d : D( this after[1,2] $b, this after[2,3] $c )
    not E( this after[1,10] $d )
then
    // do something
end
```

Temporal Distance Algorithm

1. Gather all temporal relationships between events
2. Create the temporal dependency graph as a dependency matrix
3. Calculate the reflexive and transitive closures
   - Floyd-Warshall algorithm: $O(n^3)$
4. Check for unbound intervals
   - Infinite time-windows
5. Calculate the maximum expiration time for each of the event types
6. Calculate necessary delay for the rules with negative patterns

Temporal Dependency Matrix



	A	B	C	D	E
A	[0, 0]	[-2, 2]	[-3, 4]	[-∞, ∞]	[-∞, ∞]
B	[-2, 2]	[0, 0]	[-∞, ∞]	[1, 2]	[-∞, ∞]
C	[-4, 3]	[-∞, ∞]	[0, 0]	[2, 3]	[-∞, ∞]
D	[-∞, ∞]	[-2, -1]	[-3, -2]	[0, 0]	[1, 10]
E	[-∞, ∞]	[-∞, ∞]	[-∞, ∞]	[-10, -1]	[0, 0]

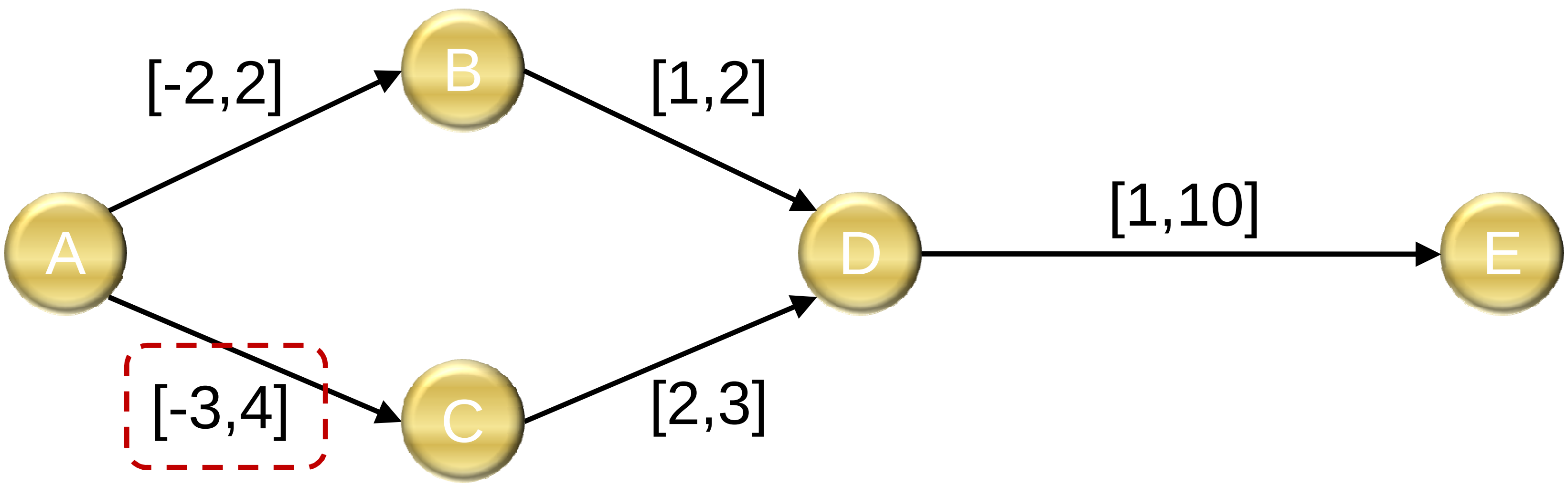
Temporal Dependency Matrix

	A	B	C	D	E
A	[0, 0]	[-2, 2]	[-3, 4]	[-∞, ∞]	[-∞, ∞]
B	[-2, 2]	[0, 0]	[-∞, ∞]	[1, 2]	[-∞, ∞]
C	[-4, 3]	[-∞, ∞]	[0, 0]	[2, 3]	[-∞, ∞]
D	[-∞, ∞]	[-2, -1]	[-3, -2]	[0, 0]	[1, 10]
E	[-∞, ∞]	[-∞, ∞]	[-∞, ∞]	[-10, -1]	[0, 0]

Transitive Closure

	A	B	C	D	E
A	[0, 0]	[-2, 2]	[-3, 2]	[-1, 4]	[0, 14]
B	[-2, 2]	[0, 0]	[-2, 0]	[1, 2]	[2, 12]
C	[-2, 3]	[0, 2]	[0, 0]	[2, 3]	[3, 13]
D	[-4, 1]	[-2, -1]	[-3, -2]	[0, 0]	[1, 10]
E	[-14, 0]	[-12, -2]	[-13, -3]	[-10,-1]	[0, 0]

Temporal Dependency Matrix



	A	B	C	D	E
A	[0, 0]	[-2, 2]	[-3, 2]	[-1, 4]	[0, 14]
B	[-2, 2]	[0, 0]	[-2, 0]	[1, 2]	[2, 12]
C	[-2, 3]	[0, 2]	[0, 0]	[2, 3]	[3, 13]
D	[-4, 1]	[-2, -1]	[-3, -2]	[0, 0]	[1, 10]
E	[-14, 0]	[-12, -2]	[-13, -3]	[-10,-1]	[0, 0]

Some references

- **Teodosiu, Dan and Pollak, Günter**. *Discarding Unused Temporal Information in a Production System*.

Q&A

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

Adam Mollenkopf
adam.mollenkopf@fedex.com
Strategic Technologist
FedEx Custom Critical

Edson Tirelli
etirelli@redhat.com
Lead CEP Engineer
JBoss, a Division of Red Hat