JBoss WORLD CHICAGO 2009

Applying Drools Fusion Complex Event Processing (CEP) for Real-Time Intelligence

Adam Mollenkopf
Strategic Technologist
FedEx Custom Critical

Edson Tirelli
Sr. Software Engineer
JBoss Enterprise Middleware

Sept. 2, 2009

Applying Drools Fusion
Agenda

- **Intro to Complex Event Processing (CEP)**
- **Stateful Rules Engine + CEP = Real-Time Intelligence**
  - Drools Expert + Drools Fusion
- **CEP Applied – FedEx Custom Critical Case Studies**
  - Demonstration, Architecture Review, and Code Walk-Through

Complex Event Processing
Defined

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

Event Processing
Typical Scenarios

Usually receive large volume of events, but only a small percentage are of **real interest**.

Individual Events are usually not important. The system is typically more concerned about **patterns** of **related events**, their **relationships**, and what can be **inferred** from them.

Characteristics of a CEP Engine
Event Processing

Support processing high volume **Streams** of Events.

Typically fed from **SOA endpoints** such as JMS Queues/Topics, Web Service calls, Databases, flat files, or sockets.

An **Event** is a record of state change. It is something that already happened, and the past cannot be changed, events are immutables.

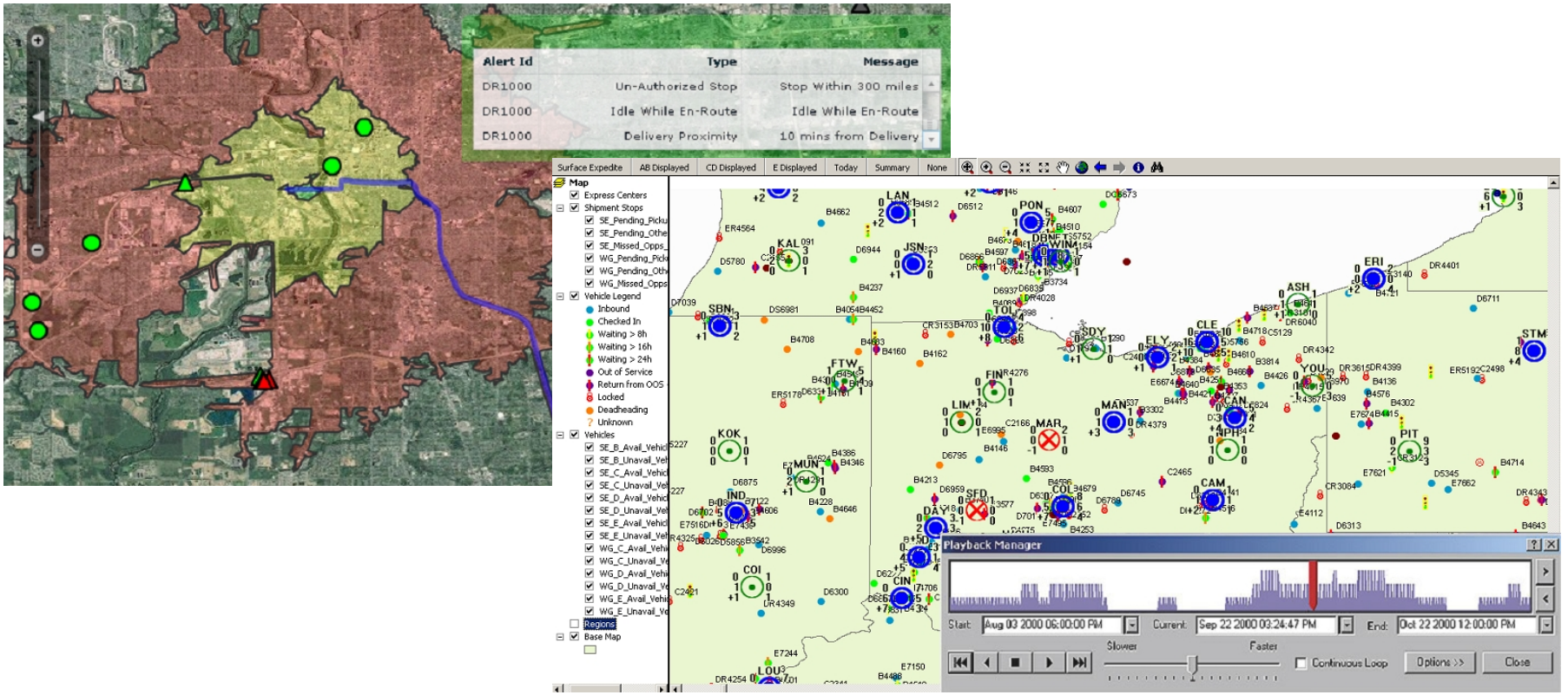
Drools Fusion – CEP Applied

Stateful Rules Engine + CEP = Real-Time Intelligence

FedEx Custom Critical Case Studies

Dispatch Eligibility Capacity Allocation Management

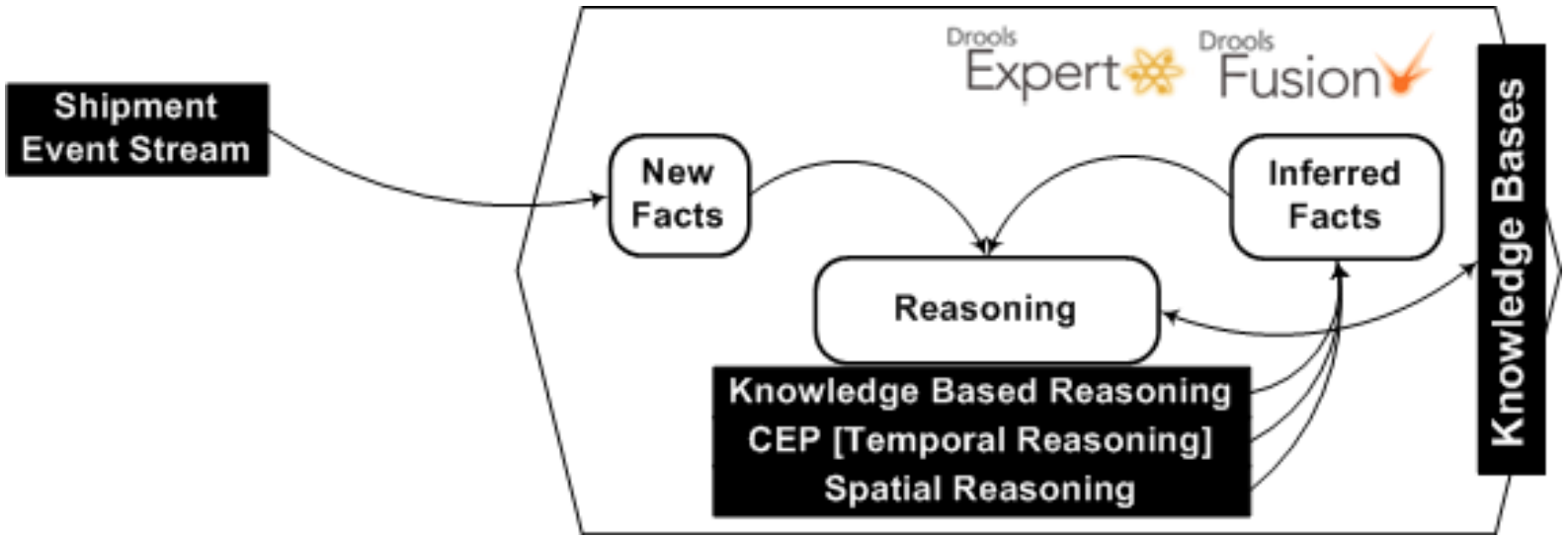
Accurate-ETA En-Route Tracking



JBoss WORLD CHICAGO 2009

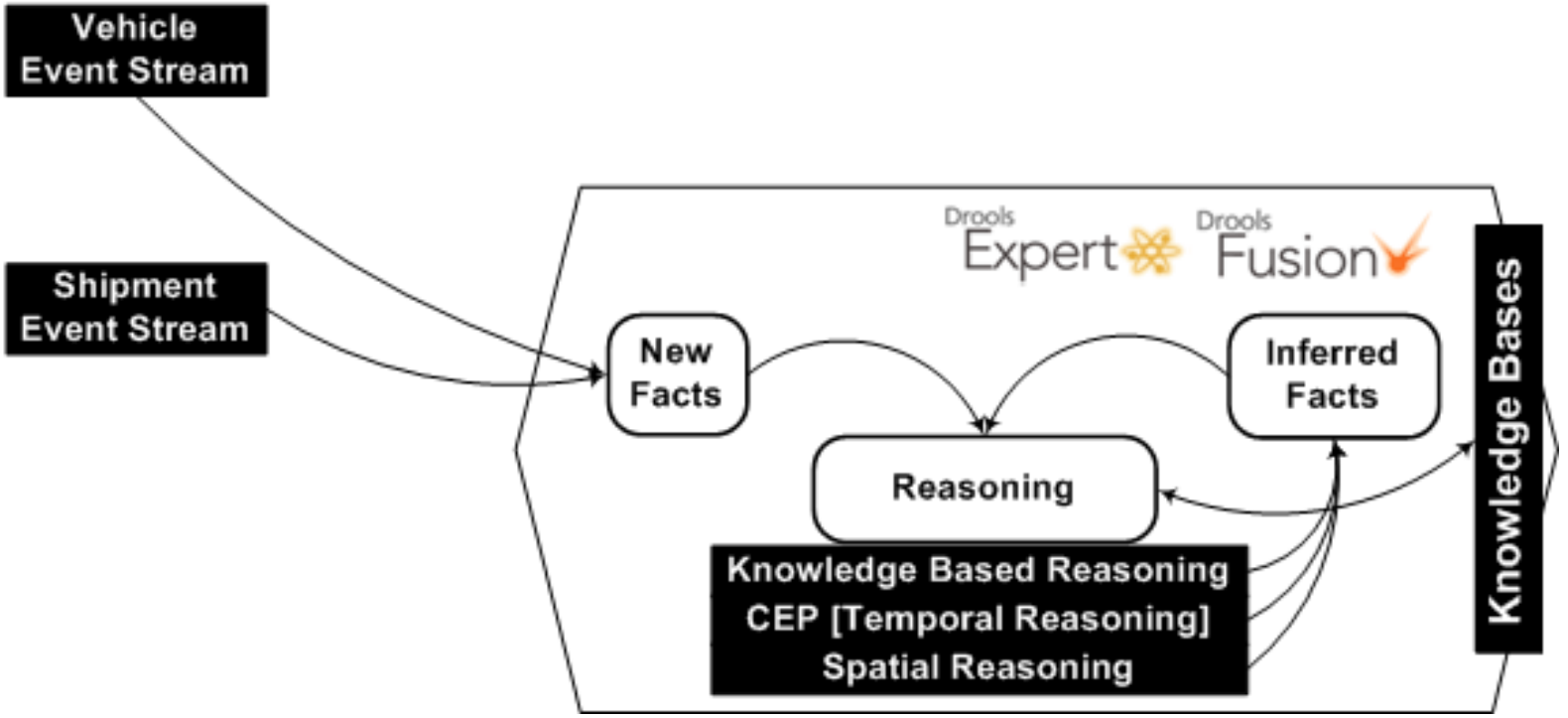
FedEx Custom Critical En-Route Tracking

CEP Applied



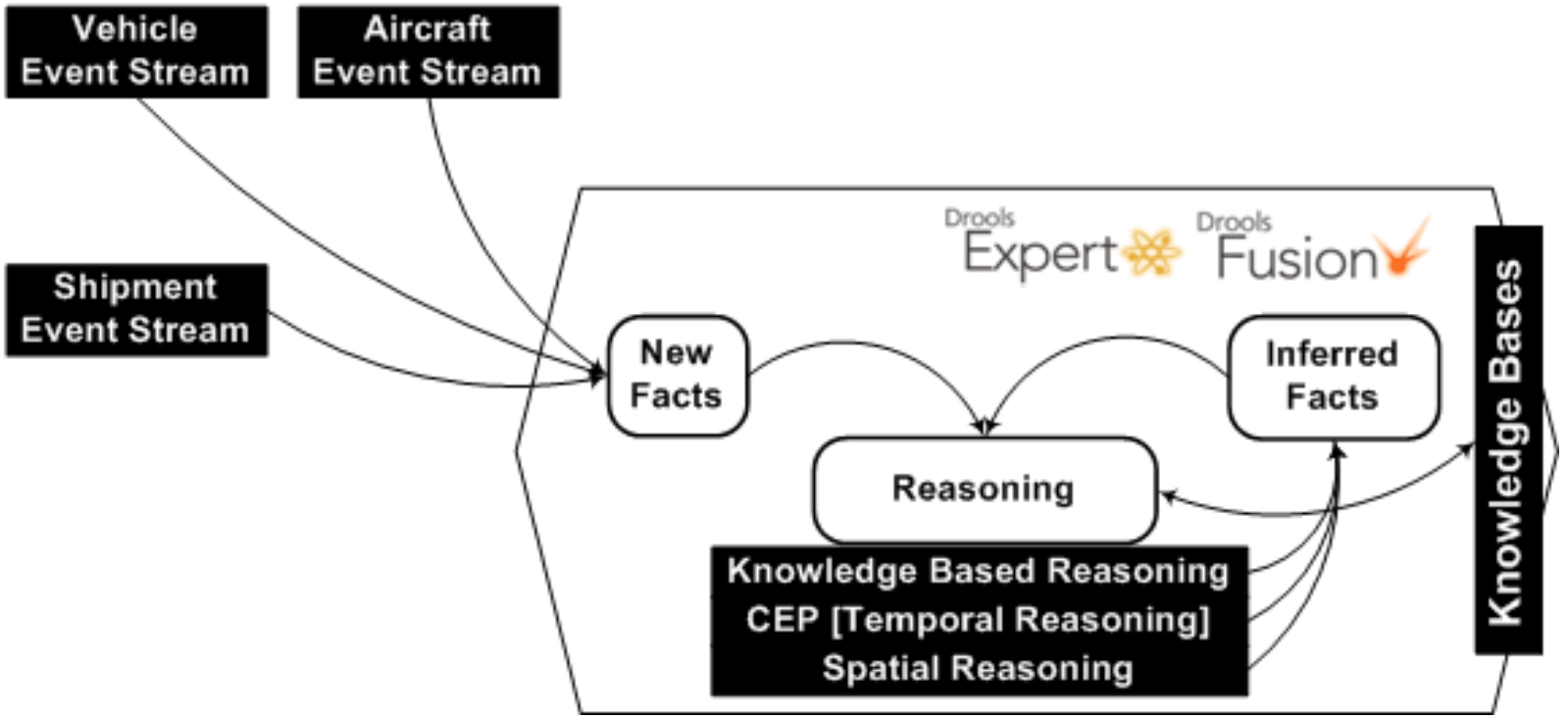
FedEx Custom Critical En-Route Tracking

CEP Applied



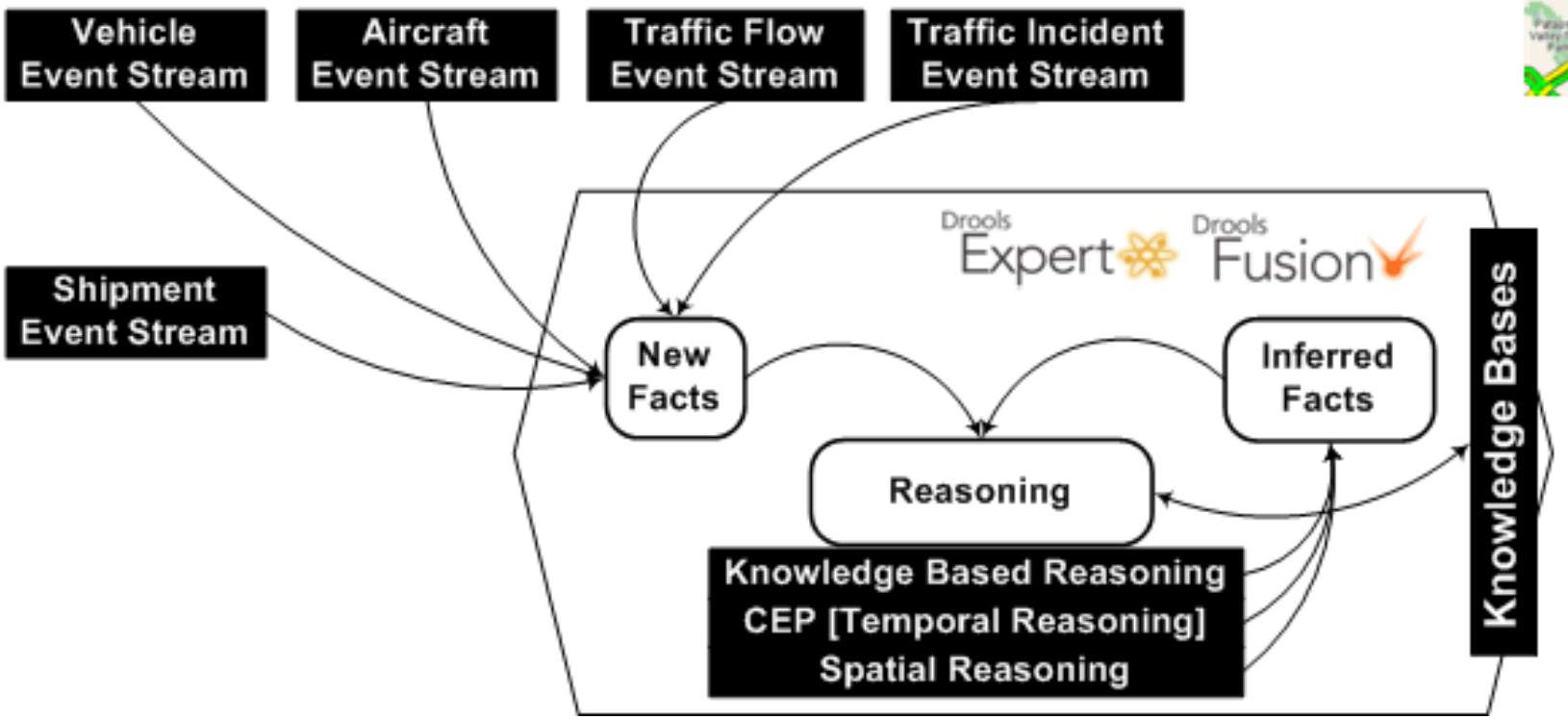
FedEx Custom Critical En-Route Tracking

CEP Applied



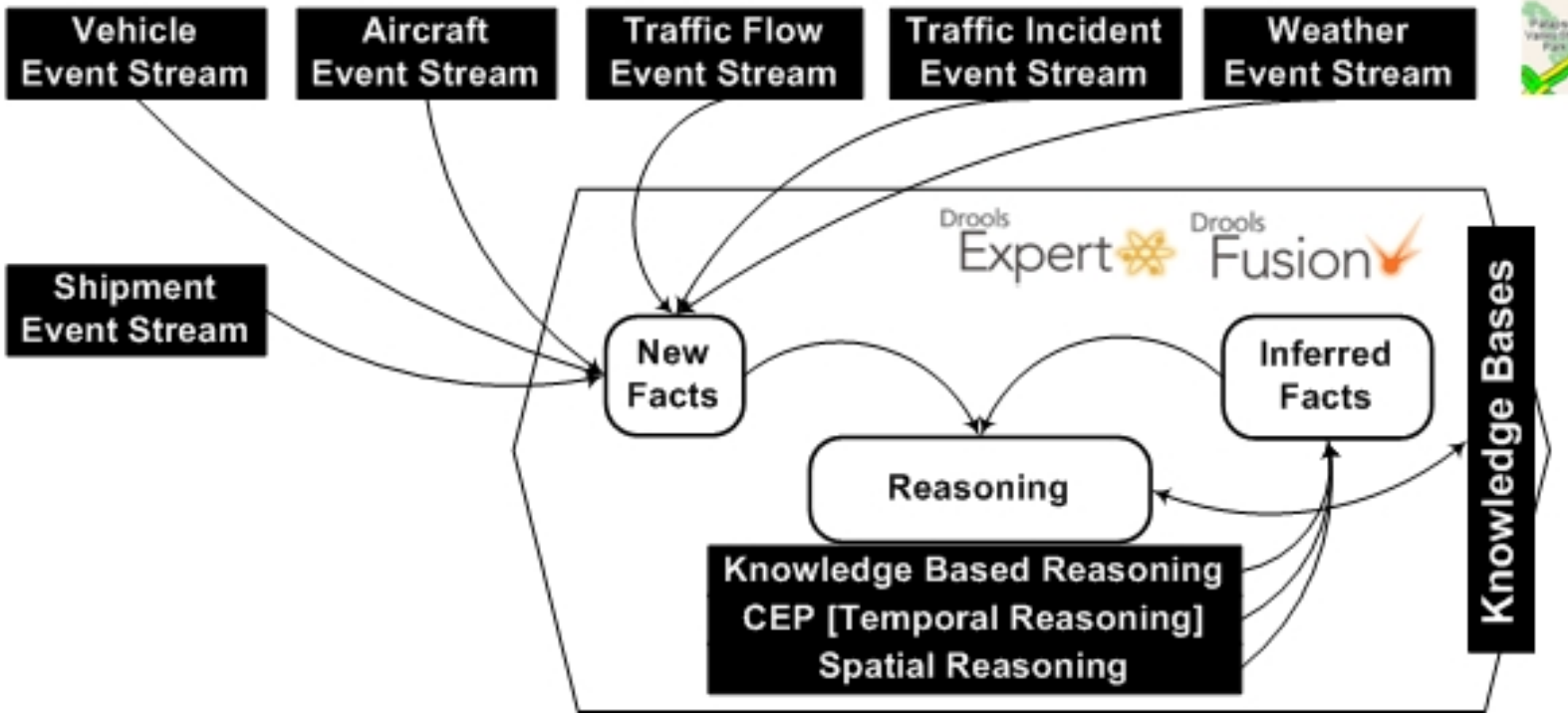
FedEx Custom Critical En-Route Tracking

CEP Applied



FedEx Custom Critical En-Route Tracking

CEP Applied



Guvnor

Expert

Fusion

Flow

Conditional Expressions
from

‘from’ can work on any expression, not just a nested field

Using ‘from’ with Hibernate Named Queries

```
rule “Find Vehicles for a given zip code"
when
    $zipCode : ZipCode()
    Vehicle() from $hibernate.getNamedQuery( “FindVehicles” )
                                .setParameters( [ “zipCode” : $zipCode ])
                                .list()
then
    …
end
```

Hibernate session

Conditional Expressions
accumulate

Accumulating Values

```
rule “accumulate"
when
   $acc : Number( intValue > 100 ) from accumulate
                                       ( Vehicle( kind == “E”, $s : shipmentCount )
                                         sum( $s ) )
then
   print “Tractor/Trailer shipment count is “ + $acc.sum;
end
```

Conditional Expressions
accumulate

Patterns and Conditional Expressions can be chained with

Patterns and CE Chaining with ‘from’

```
rule “collect"
when
   $zipCode : ZipCode()
   $acc : Number( intValue > 100 ) from accumulate
                        ( Bus( kind == “E”, $s : shipmentCount )
                          from $hibernate.getNamedQuery( “FindVehicles” )
                                .setParameters( [ “zipCode” : $zipCode ])
                                .list(), sum( $s ) )
then
   print “Tractor/Trailer shipment count for “ + $zipCode + “ is “ + $acc.sum;
end
```

Concurrent Event Stream Processing

Misperception 1: Rules Engines do not scale for CEP

Rules Engines have historically had a single point of inser

CEP allows for concurrent Streams of Events.

Rules Engine Scaling problem with CEP

```
ruleEngineSession.insert( event );    // Single point of entry
```

```
rule "Process Scheduled Pickup"
when
    $c : Customer( type == "VIP" )
    ScheduledPickupEvent( customer == $c )
then
    …
end
```

Patterns evaluate facts sequentially in a single thread.

Concurrent Event Stream Processing

entry-point = Scalable

Concurrent Event Stream Processing is made possible via

Using entry-point in a Rules-Engine for CEP

```
EntryPoint ep = session.getEntryPoint( “ShipmentEventsEP” );
ep.insert( event );    // Now we can insert different streams concurrently
```

```
rule “Process Scheduled Pickup from Entry Point"
when
    $c : Customer( type == “VIP” )
    ScheduledPickupEvent( customer == $c ) from entry-point “ShipmentEP”
then
    …
end
```

When not specified the “default” entry-point is used

Patterns can now optionally specify their entry-point

Automatic Life-Cycle Management

@role(event)

Fact life-cycles must be managed by the user, so retraction

Event life-cycles are automatically managed.

Declaring a type as an Event

```
declare VehiclePositionEvent
    @role( event )
end;    // will be retracted when it is no longer needed
```

```
declare VehiclePositionEvent
    @role( event )
    @timestamp( timestampAttr )
    vehicleId : String
    longitude : double
    latitude : double
    timestampAttr : long
end;
```

Temporal Operators

Reasoning over Time

Misperception 2: **Rule Engines do not have a rich enough s**

Temporal Operator ‘after’ Detection

```
rule “Confirm Pickup occurs within Pickup standard"
when
    $c : Customer( type == “VIP” )
    $spe : ScheduledPickupEvent( customer == $c )
        from entry-point “ShipmentEP”
    PickupEvent( relatedEvent == $spe.id, this after[0m, 30m] $spe )
        from entry-point “VehicleEP”
then
    …
end
```

PickupEvent must occur between
0 and 30 minutes ‘after’
ScheduledPickupEvent

Temporal Operators
Reasoning over Time

Temporal Operator Not ‘after’ Detection

```
rule “Detect Pickups that have not occurred within Pickup standadr"
when
    $c : Customer( type == “VIP” )
    $spe : ScheduledPickupEvent( customer == $c )
                        from entry-point “ShipmentEP”
    not PickupEvent( relatedEvent == $spe.id, this after[1s, 10s] $spe )
                        from entry-point “VehicleEP”
then
    …
end
```

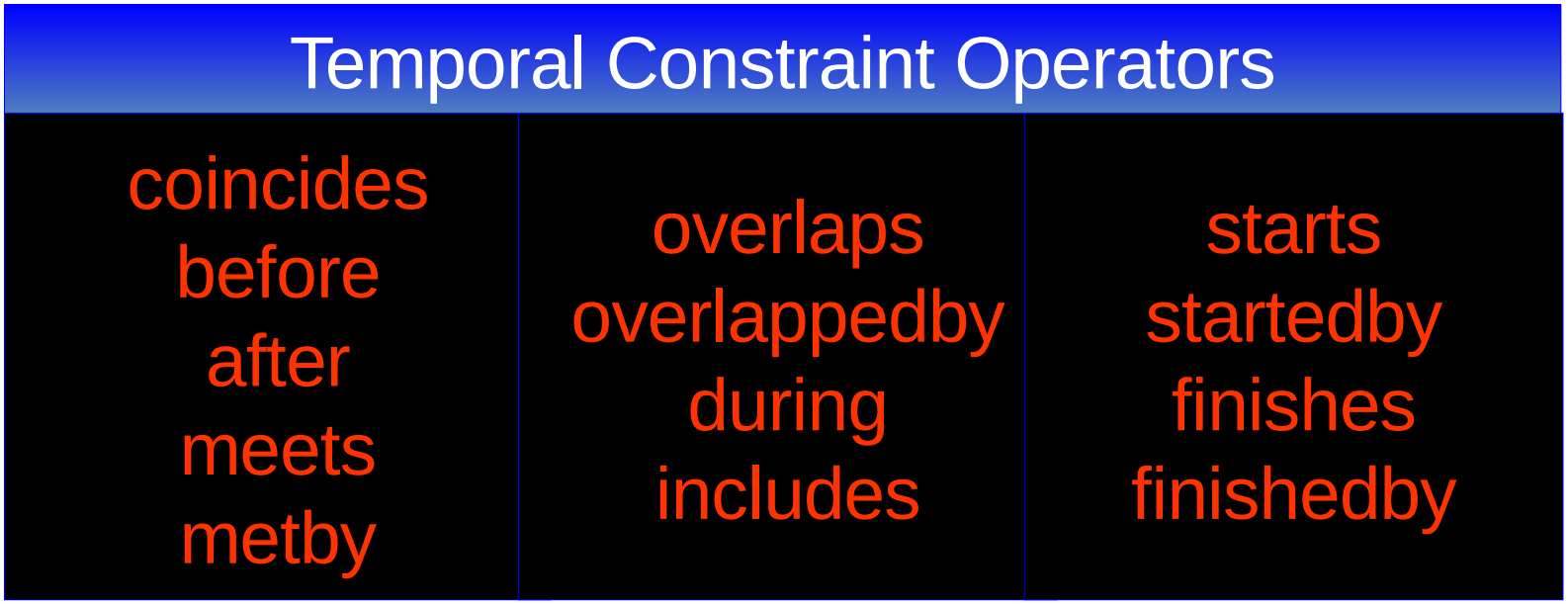
Existing Drools ‘not’ Conditional Elements
can be used to detect non-occurrence of events

Temporal Operators

Reasoning over Time

Allow for detection, correlation, aggregation, and composi

Temporal Constraint operators express relationship betwe



Temporal Operators

Reasoning over Time

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

Temporal Reasoning
Sliding Time Windows

Misperception 3: **Rule Engines react to events happening n**

```
TemperatureRead( vehicleId = “E1000” )  over window:time ( 10m )
TemperatureRead( vehicleId = “E1000” )  over window:length ( 10 )
```

Temporal Reasoning
Aggregations

Misperception 4: **Rules Engines do not deal with aggregatio**

```
rule “Average temperature reading for vehicle E1000 over last 10 minutes"
when
    $n : Number( intValue > 8 )
                    from accumulate ( $tr : TemperatureRead( vehicleId = “E1000”
)
                    over window:time( 10m ),
                    average( $tr.temperatureValue ) )
then
    …
end
```

Applying CEP at FedEx Custom Critical

Declaring point-in-time Vehicle Event

```
import com.fedex.enroutetracking.VehicleSensors;
declare VehicleSensors
    @role( event )
    @timestamp ( lastEventTime.time )
    @expires ( 1h30m ) // of the form:  [#d] [#h] [#m] [#s] [#[ms]]
end
```

Declaring interval-based TrafficIncident Event

```
import com.fedex.tracking.traffic.TrafficIncident;
declare TrafficIncident
    @role( event )
    @timestamp ( startTime )
    @duration ( incidentDuration )    // in milliseconds
end
```

Applying CEP at FedEx Custom Critical

Producing Events to an Entry-Point

```
KnowledgeBase kb = readKnowledgeBase();
StatefulKnowledgeSession ks = kb.newStatefulKnowledgeSession();
WorkingMemoryEntryPoint ep = ks.getWorkingMemoryEntryPoint(“VehicleEP”);
Vehicle vehicle = …;
vehicleEP.insert(vehicle);
```

Consuming Events from an Entry-Point

```
rule "Revise Stop ETAs based on new Vehicle Position"
when
   $vehicle : Vehicle ( $vid : vehicleId, $pos : position,
                        status == “EnRouteToStop” ) from entry-point ”VehicleEP"
   $stops : Stops ( vehicleId == $vid )
then
   $stops = EtaCalculator.reviseStopETAs( $vehicle, $stops );
   update( $stops );
end
```

Applying CEP at FedEx Custom Critical

Detect and react to patterns in events for **Sliding Windows** of

Detecting Idle (Slow Moving) Vehicles

```
rule "Speed Idle En-Route Alert"
when
    $stats : SensorStatistics( $vid : vehicleId )
    $alerts : Alerts( vehicle == $vid )
    $vehicle : Vehicle ( vehicleId == $vid, inStopProximity == false )
    Number ( $avgSpeedMph : floatValue < 15 )
        from accumulate (
            VehicleSensors( vehicleId == $vid, inStopProximity == false,
                                $speedMph : speedMph )
            over window:time ( 15m ) from entry-point "VehicleEP",
            average ( $speedMph ) )
then
    $alerts.addAlert( new Alert (…) );
    update ( $alerts );
end
```

Accumulator functions: sum, count, min, max collect, [custom defined]

Applying CEP at FedEx Custom Critical

Deterministic Simulation Testing (Pseudo-Clock)

Reasoning over time requires a **Reference Clock**.

Session Clock implements the GoF Strategy Pattern.

Rules Testing requires a **controlled** environment of input rules

To Replay or Simulate scenarios it is necessary to control the **F**

Regular Execution requires a **Real-Time Clock**.

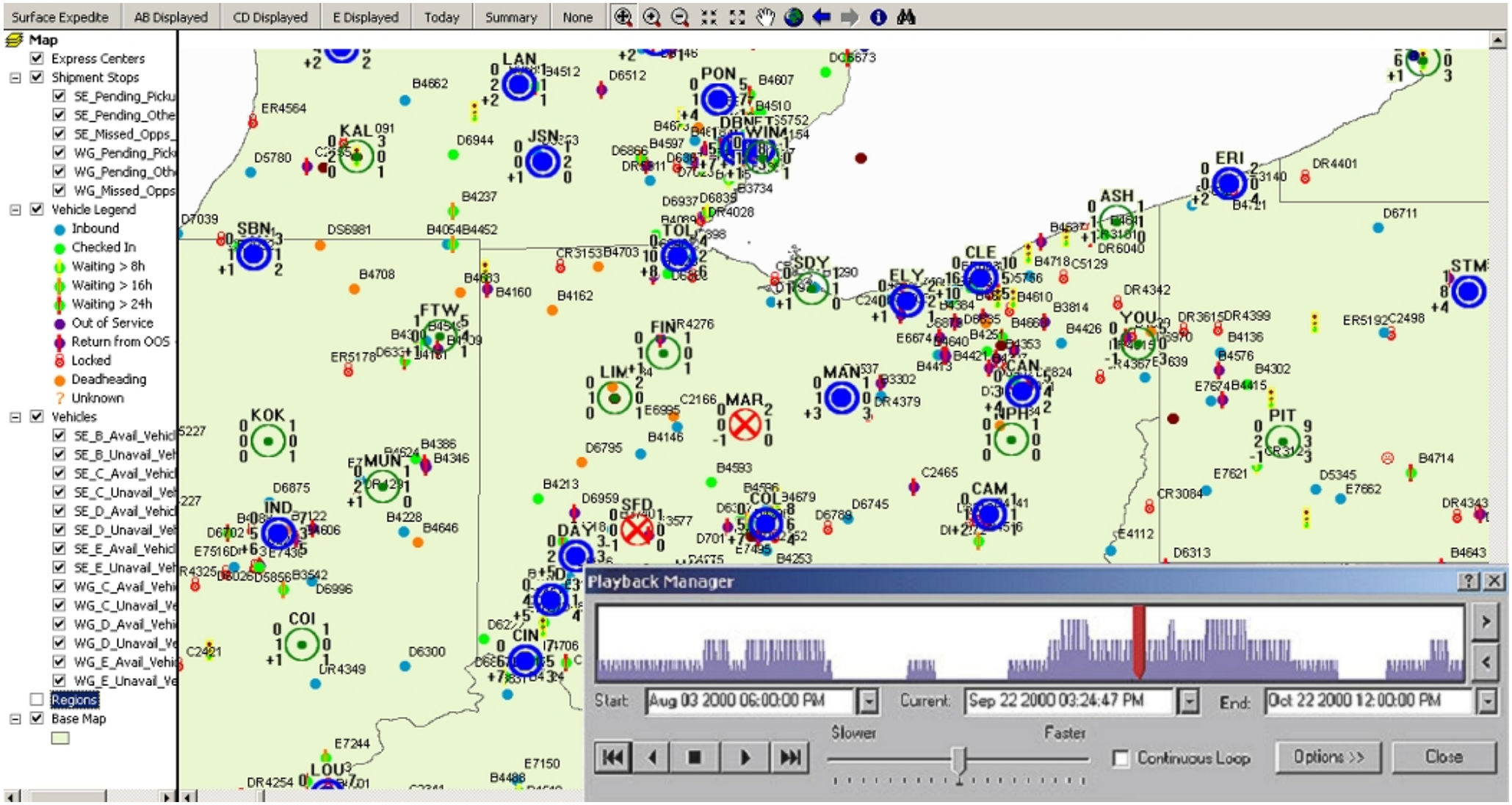
Applying CEP at FedEx Custom Critical
En-Route Tracking Case Study

En-Route Tracking Situational Awareness

Demonstration,
Architecture Review,
and Code Walk-Through

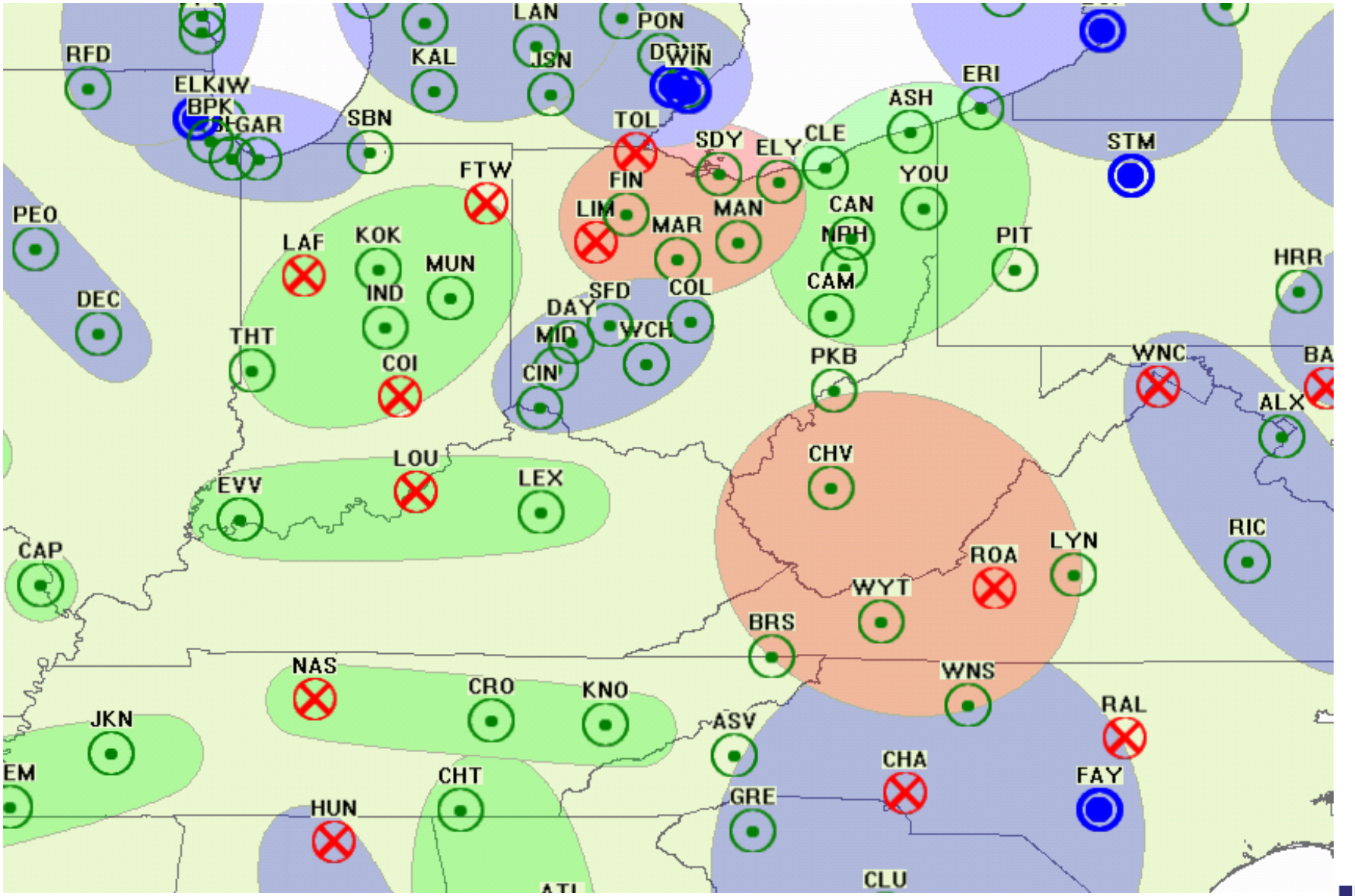
Applying CEP at FedEx Custom Critical

Capacity Allocation Management Case Study



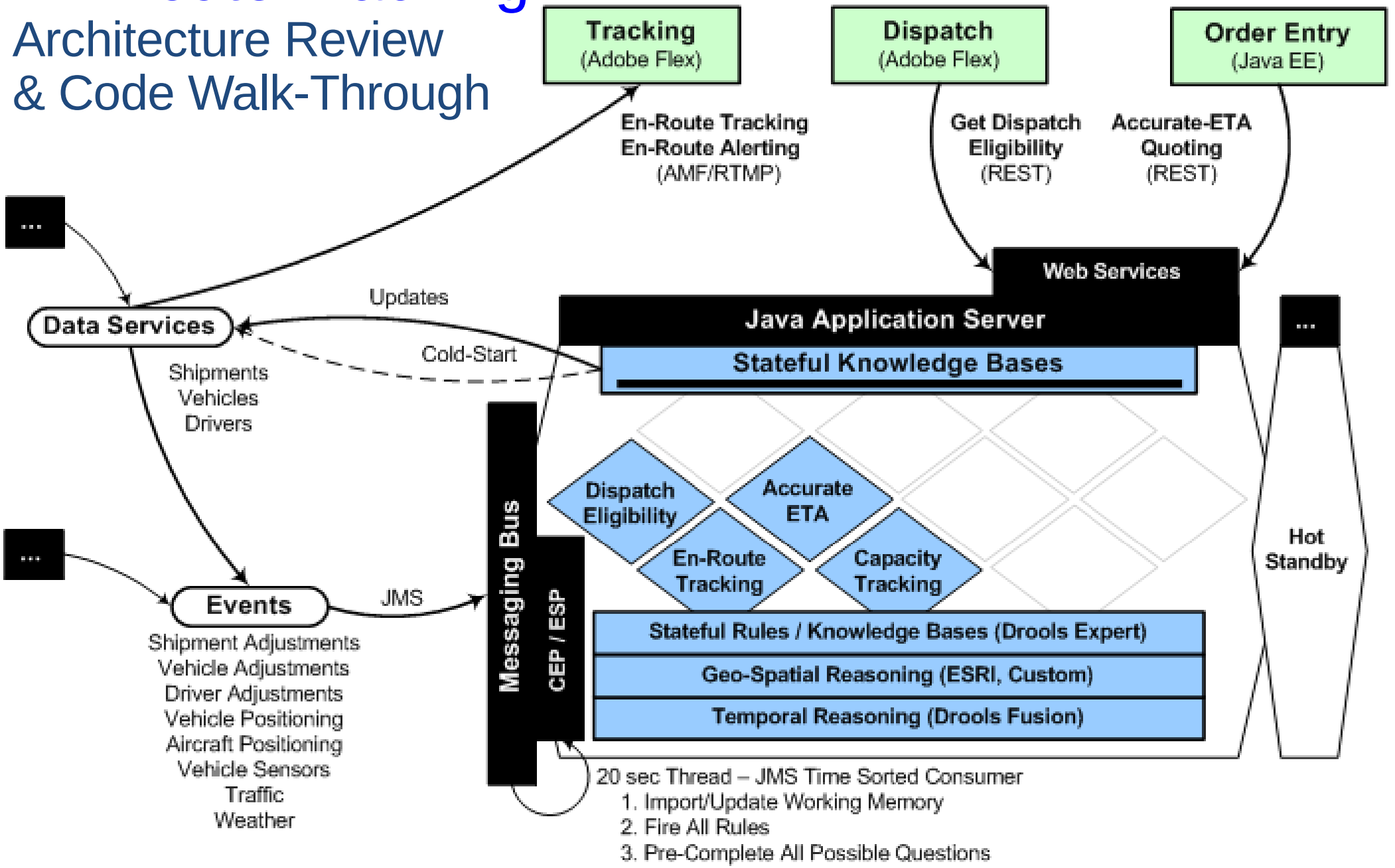
Applying CEP at FedEx Custom Critical

Capacity Allocation Management Case Study



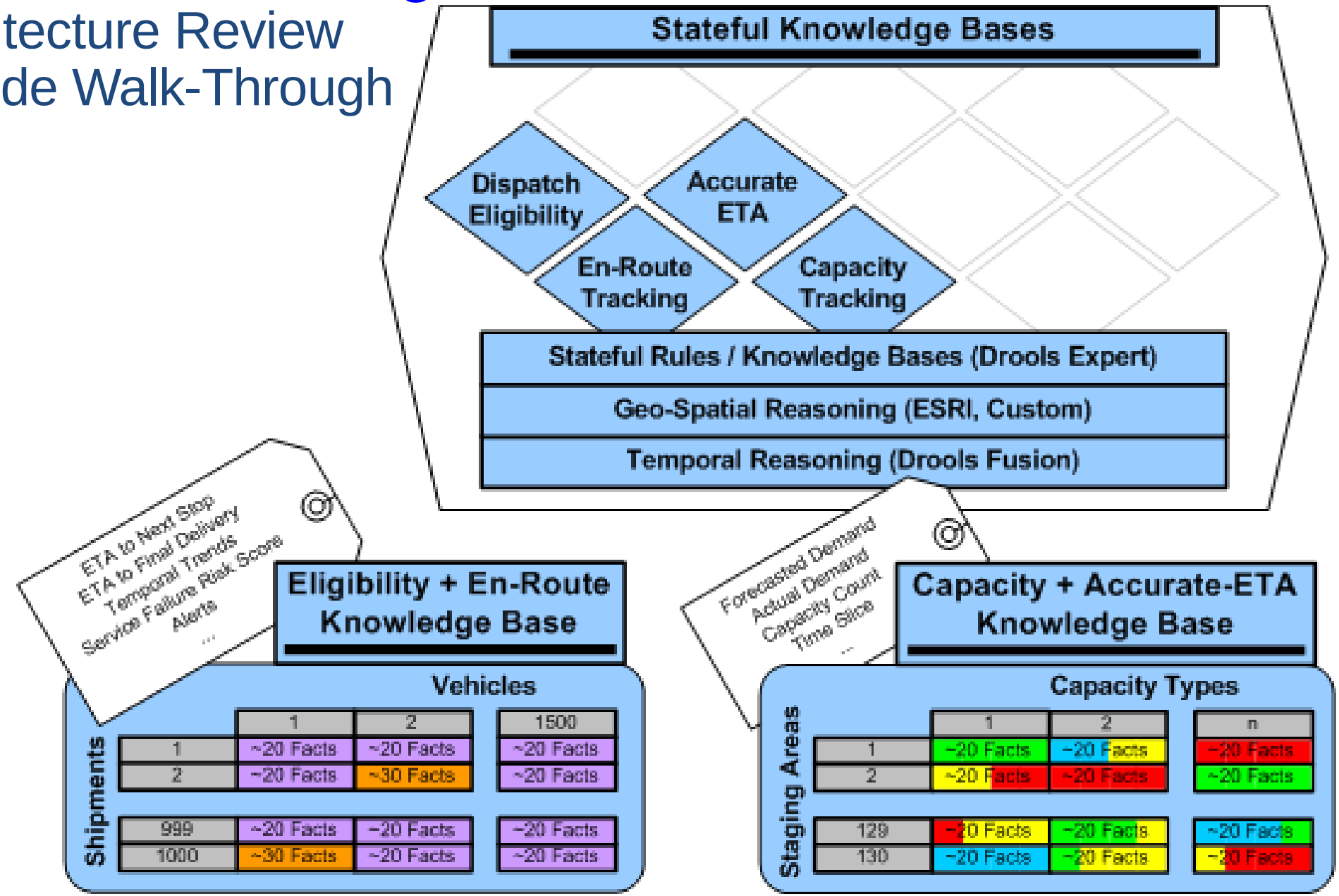
En-Route Tracking

Architecture Review & Code Walk-Through



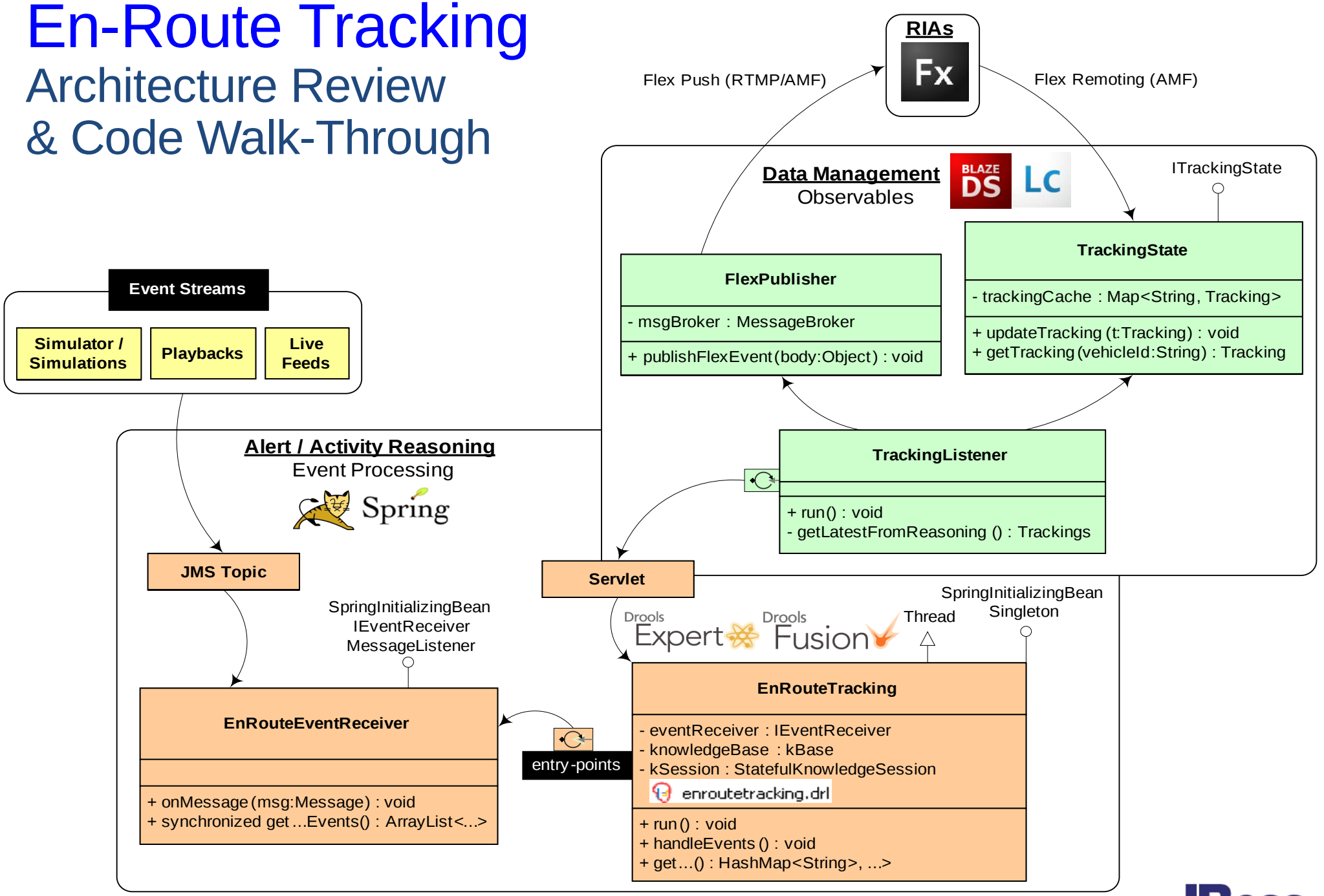
En-Route Tracking

Architecture Review & Code Walk-Through

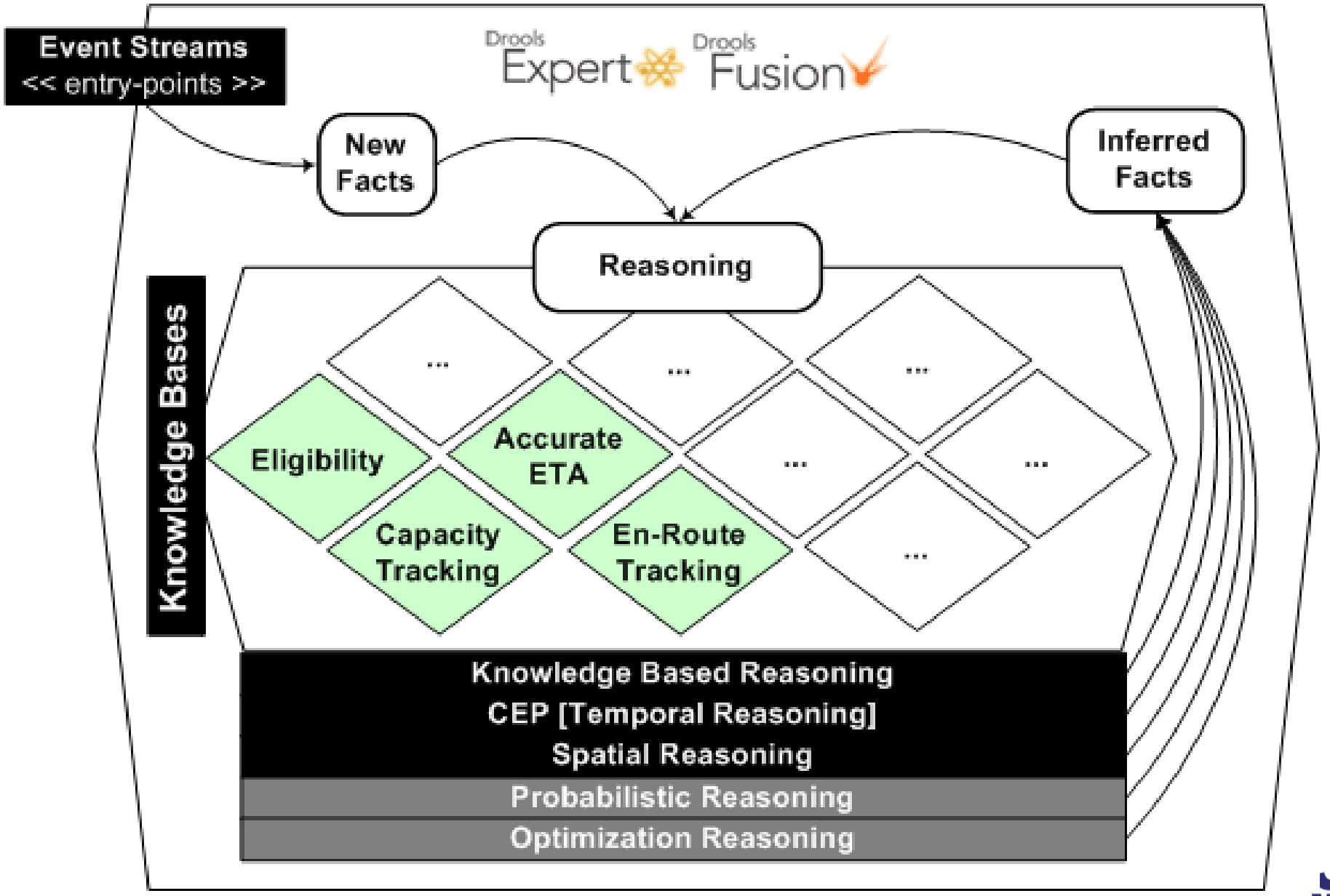


En-Route Tracking

Architecture Review & Code Walk-Through

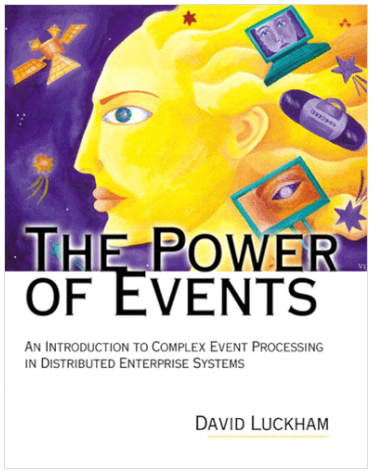


Stateful Rules Engine + CEP
= Real-Time Intelligence



Where to Learn More

Complex Event Processing



"The Power of Events" by David Luckham

See also http://complexevents.com

Vendor Documentation

Drools Fusion, Drools Expert

http://www.jboss.org.drools.documentation.html

http://www.jboss.com/drools

http://blog.athico.com

Our Contact Information:

Adam.Mollenkopf@fedex.com

ETirelli@redhat.com

QUESTIONS?

TELL US WHAT YOU THINK:
REDHAT.COM/JBOSSWORLD-SURVEY

JBoss WORLD
CHICAGO 2009

FOLLOW US:

TWITTER.COM/REDHATSUMMIT

TWEET ABOUT US:

ADD #SUMMIT AND/OR #JBOSSWORLD TO THE END OF YOUR EVENT-RELATED TWEET

presented by