CEP – Complex Event Processing

Sistemas Baseados em Regras

03/04/2012

CEP e Regras

- Um regra define um comportamento que se aplica em uma determinada *situação*
- Essa *situação* é frequentemente definida em termos de eventos complexos
- Precisamos de mecanimos para:
  - Distribuir eventos
  - Modelar eventos
  - Detectar eventos complexos
  - Etc.

Motivação

- Necessidade de adequar processamento de dados a **sistemas de tempo real** (real-time)
- “Data is on the move”
  - Dados devem ser processados *agora ou nunca*
  - E.g., notifique a pessoa caso seu voo esteja atrasado
- Informação deve encontrar os consumidores
  - Ao invés de procurarmos por informações, elas é que devem nos encontrar!

Exemplos de sistemas *real-time*

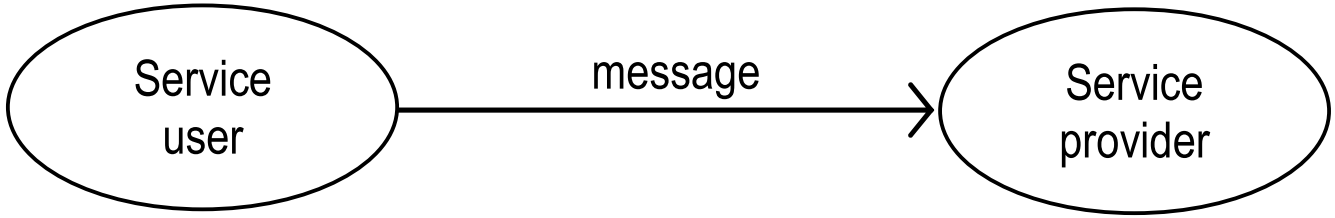
- Notifique-me caso minha bagagem não esteja embarcada no avião (antes de decolar)
- Notifique-me caso tenha amigos que estejam próximos a mim, quando eu estiver em um café
  - Combinação de eventos
- Notifique-me caso exista possibilidade de trânsito ruim em meu caminho
  - Previsão de eventos futuros

Real-time?

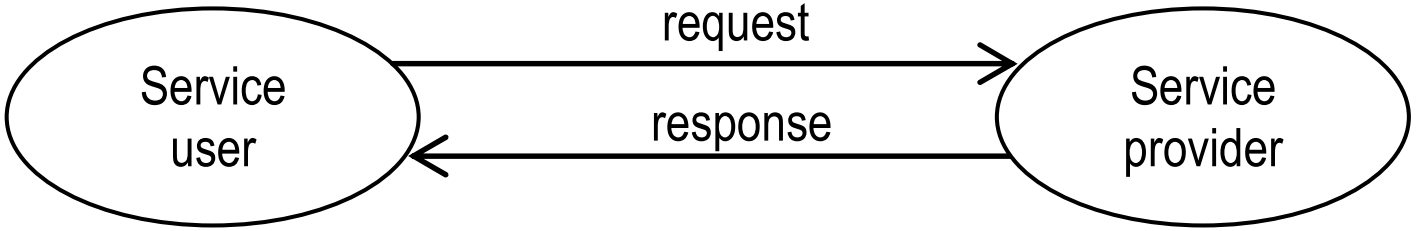
- Entretenimento
  - Reagir a Tweets
- Mercado financeiro
  - Tomar decisões financeiras em nano-segundos
- eHealth
  - Monitoramento remoto de pacientes
- Energia
  - “smart readings”

Interaction Patterns

- One-Message

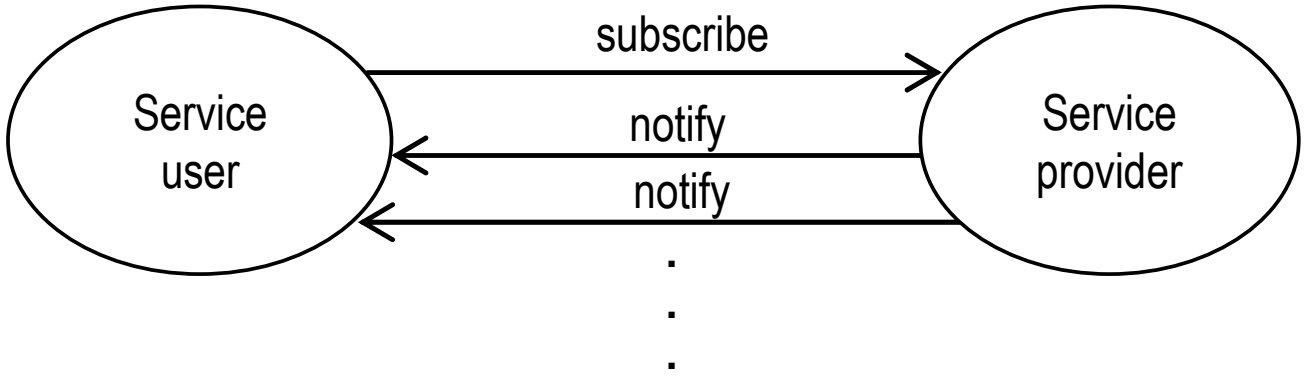


- Request-Response



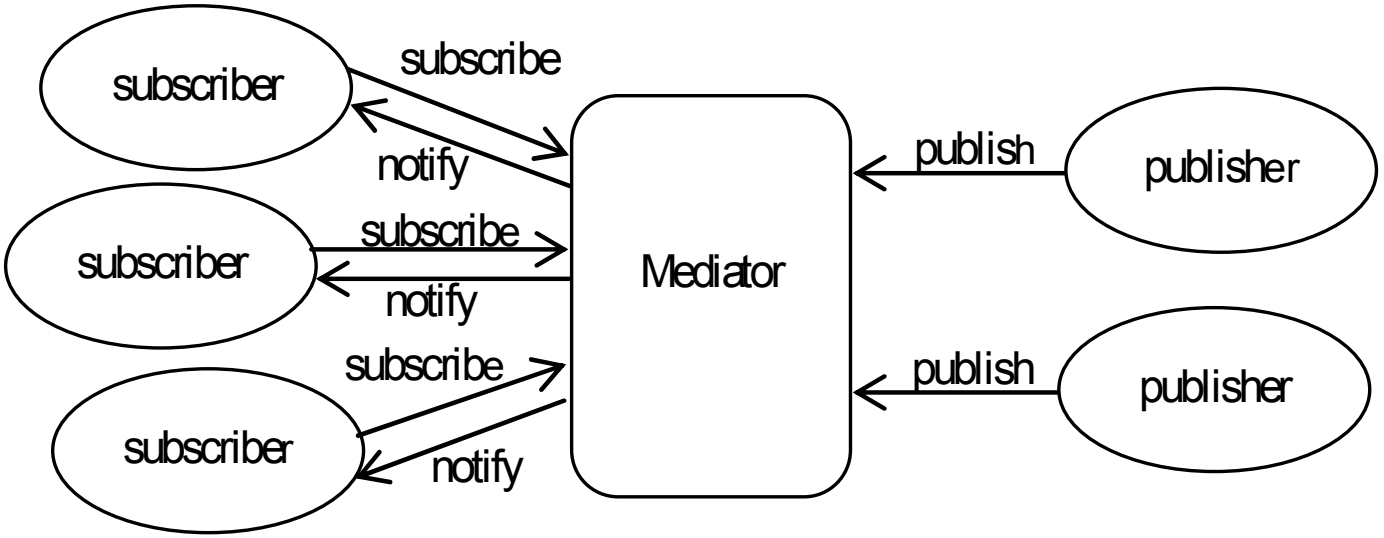
Interaction Patterns (cont.)

- Subscribe-Notity (time-based or event-based)



Interaction Patterns (cont.)

- Publish-Subscribe



Objetivos de CEP

- Resolver os problemas inerentes a sistemas *real-time* baseados em eventos
  - Muitos problemas atuais são do tipo “event-driven”
- Definir um sistema de gerenciamento de eventos

Características de sistemas baseados em eventos

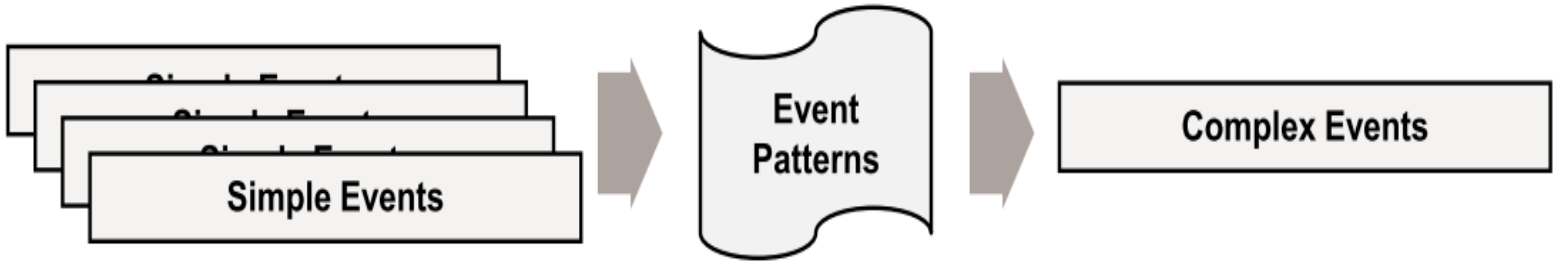
- *Real-time awareness*
- Processamento também *real-time*
- Throughput muito alto de eventos
- Combinação não-trivial de eventos
- Combinação de eventos de diferentes fontes

Evento

- "... acontecimento de interesse em um sistema ou domínio"
- Representa alguma mudança ou ocorrencia
- Pode ou não ter duração
- Informação com timestamp
- Container de informação
- *Trigger actions*

Evento Complexo

- Agregação ou derivação de eventos mais simples

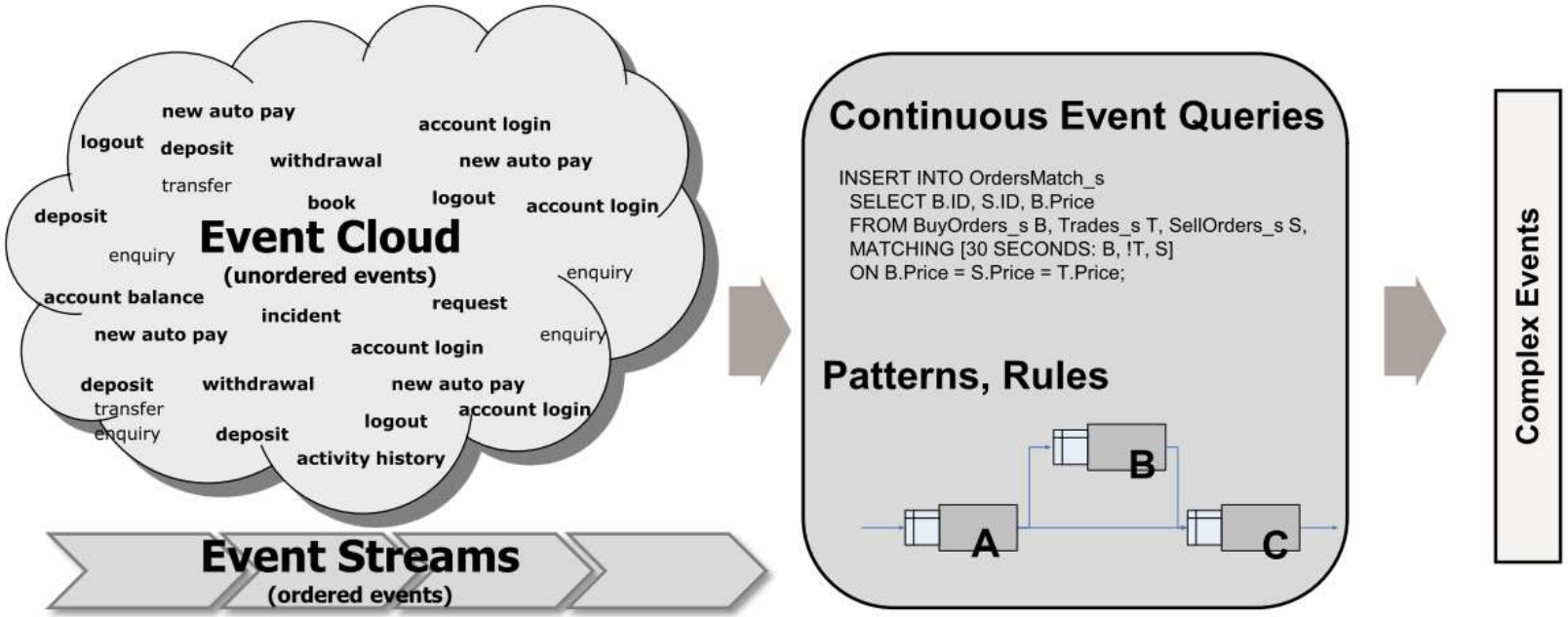


CEP

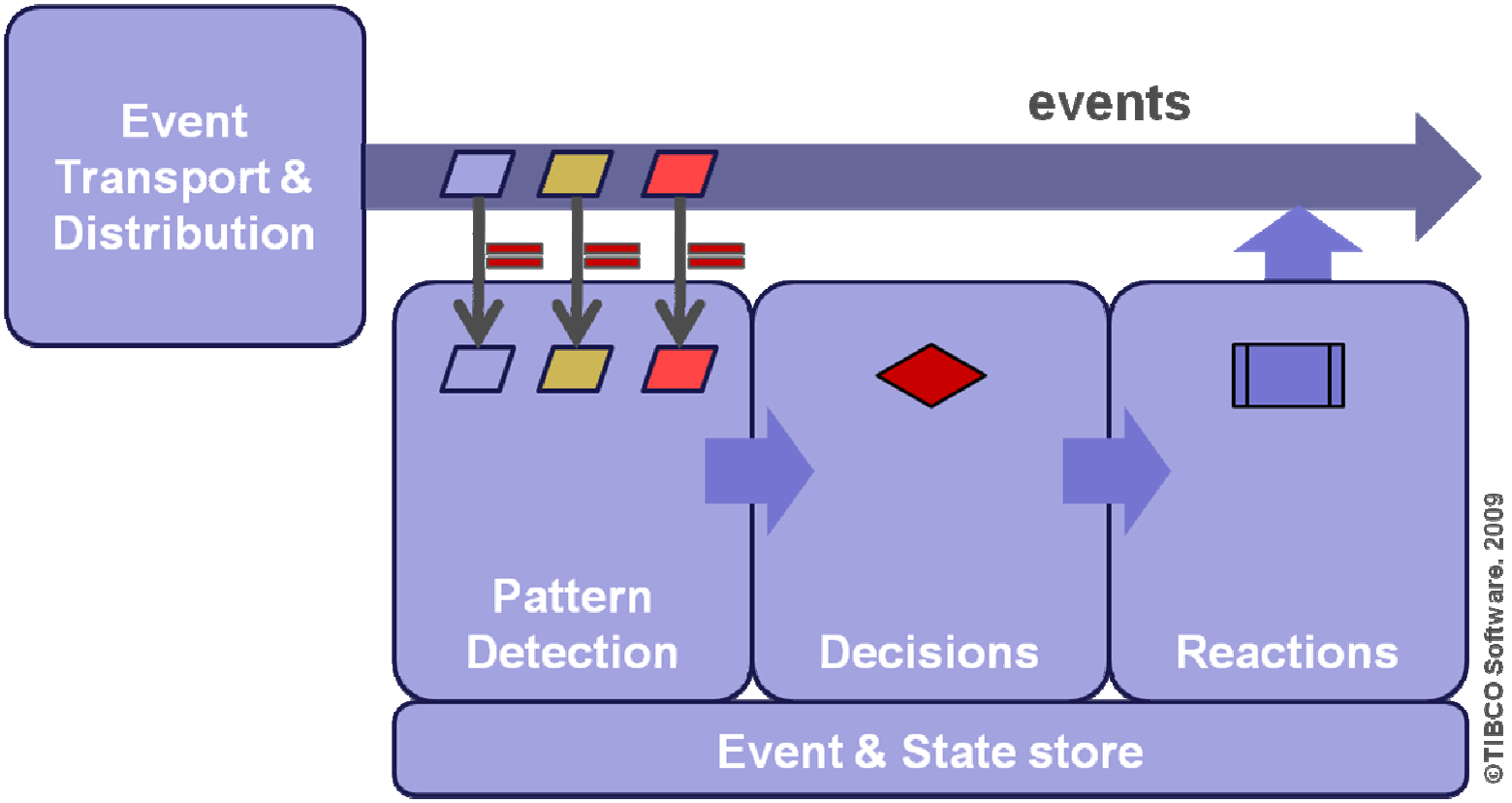
- Processamento com base em eventos
- CEP oferece suporte real-time de fluxo de eventos muito grandes
- CEP também se preocupa com o reconhecimento (*real-time* and *ahead-of-time*) de situações de interesse
  - Padrão de evento complexo (*complex event pattern*) define uma situação de interesse

CEP- cont.

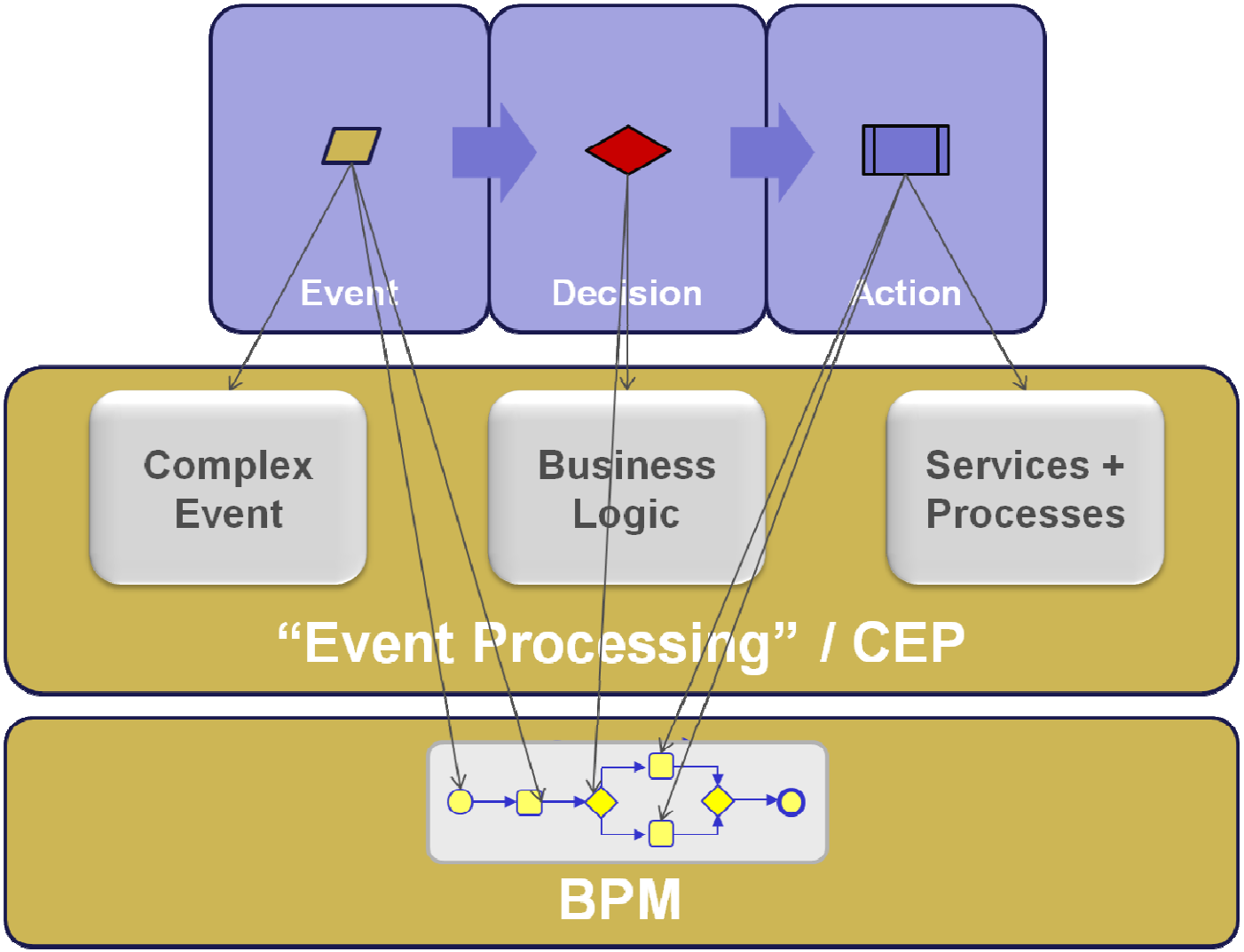
- Processamento eficiente de um número grande de eventos (near real-time)
- Detecção, previsão e exploração de eventos complexos
- Situation Awareness



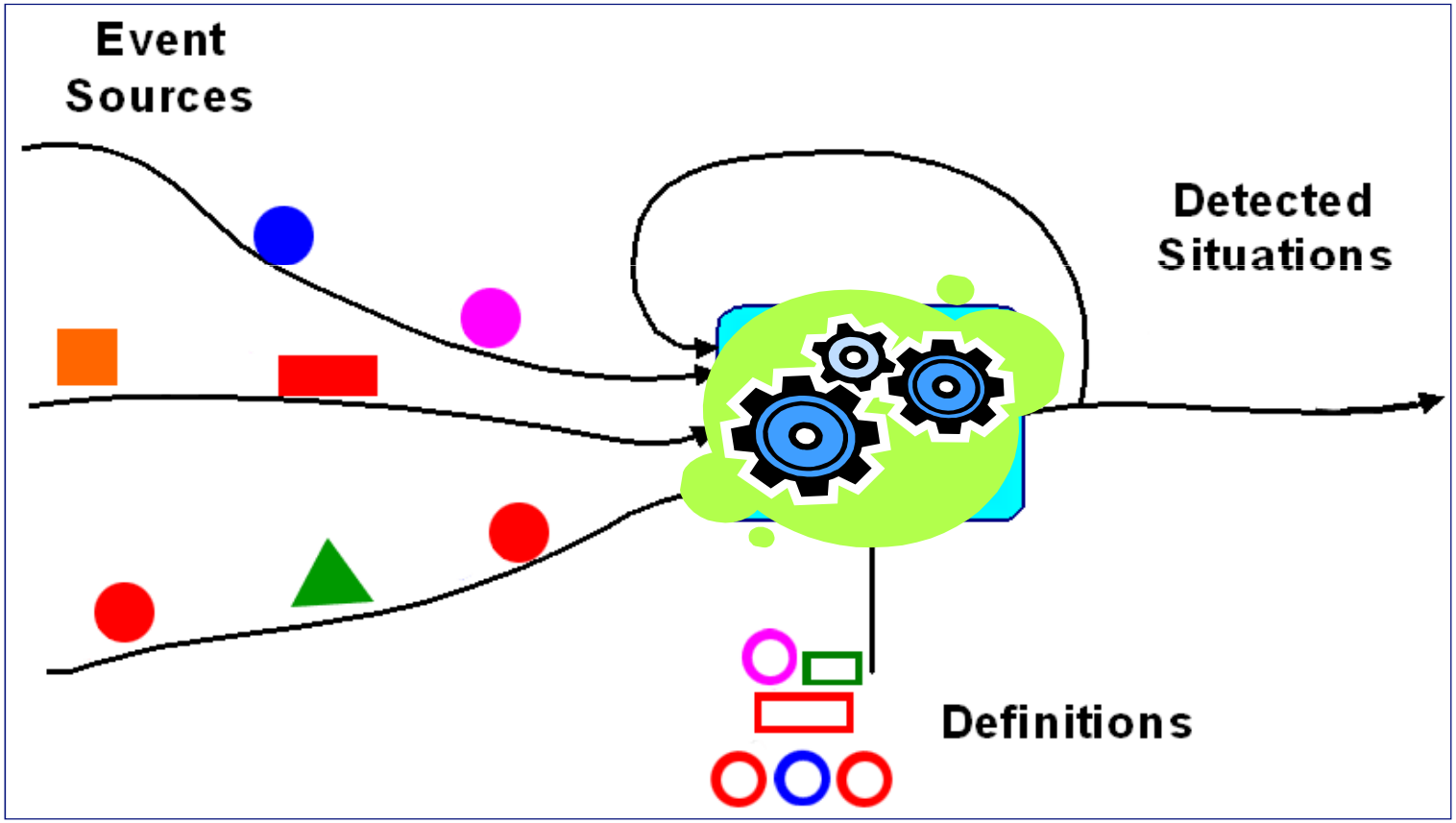
Arquitetura *event-driven*



Integração ...

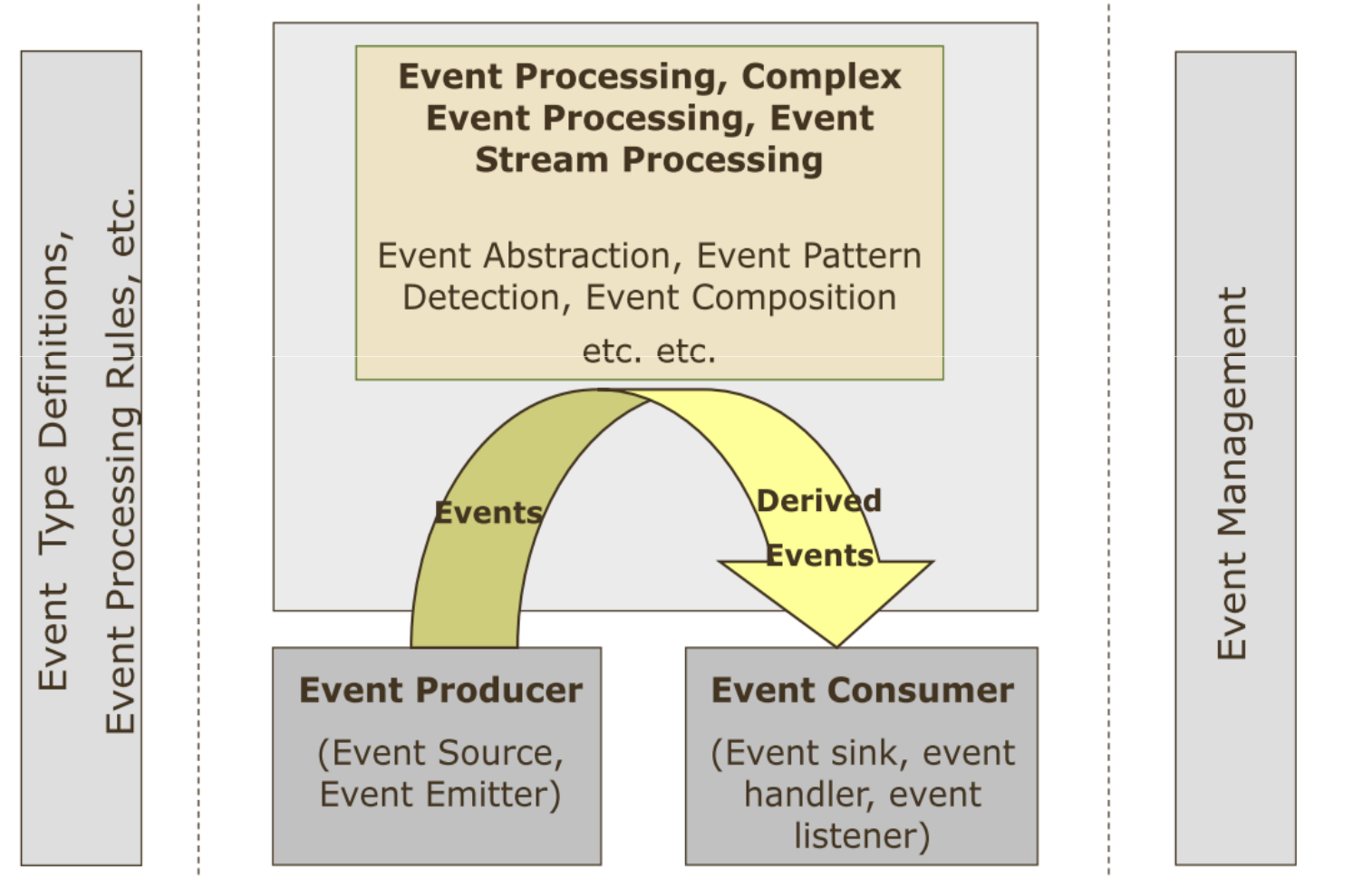

Event
Decision
Action
Complex Event
Business Logic
Services + Processes
“Event Processing” / CEP
BPM

Event Pattern Detection

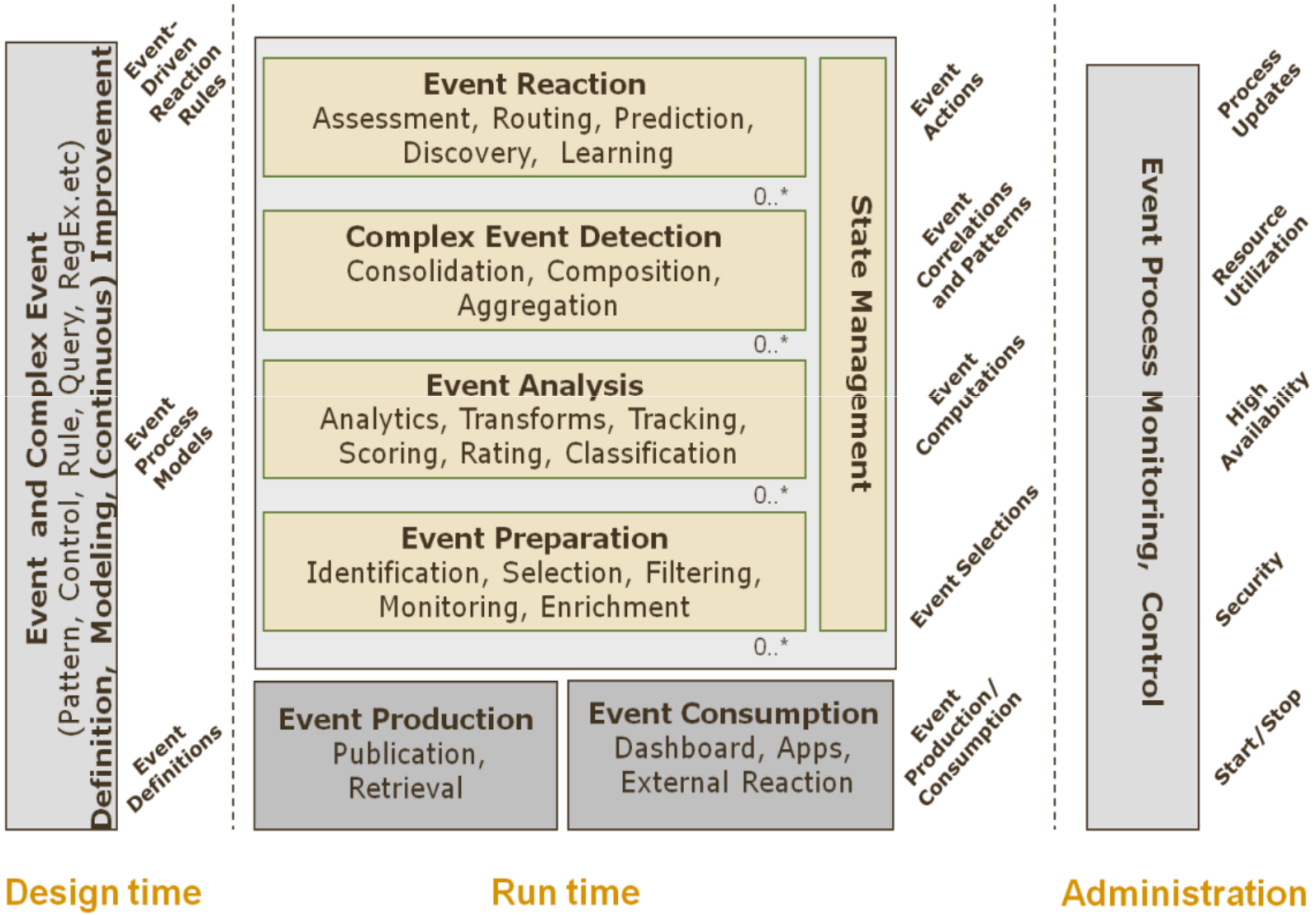


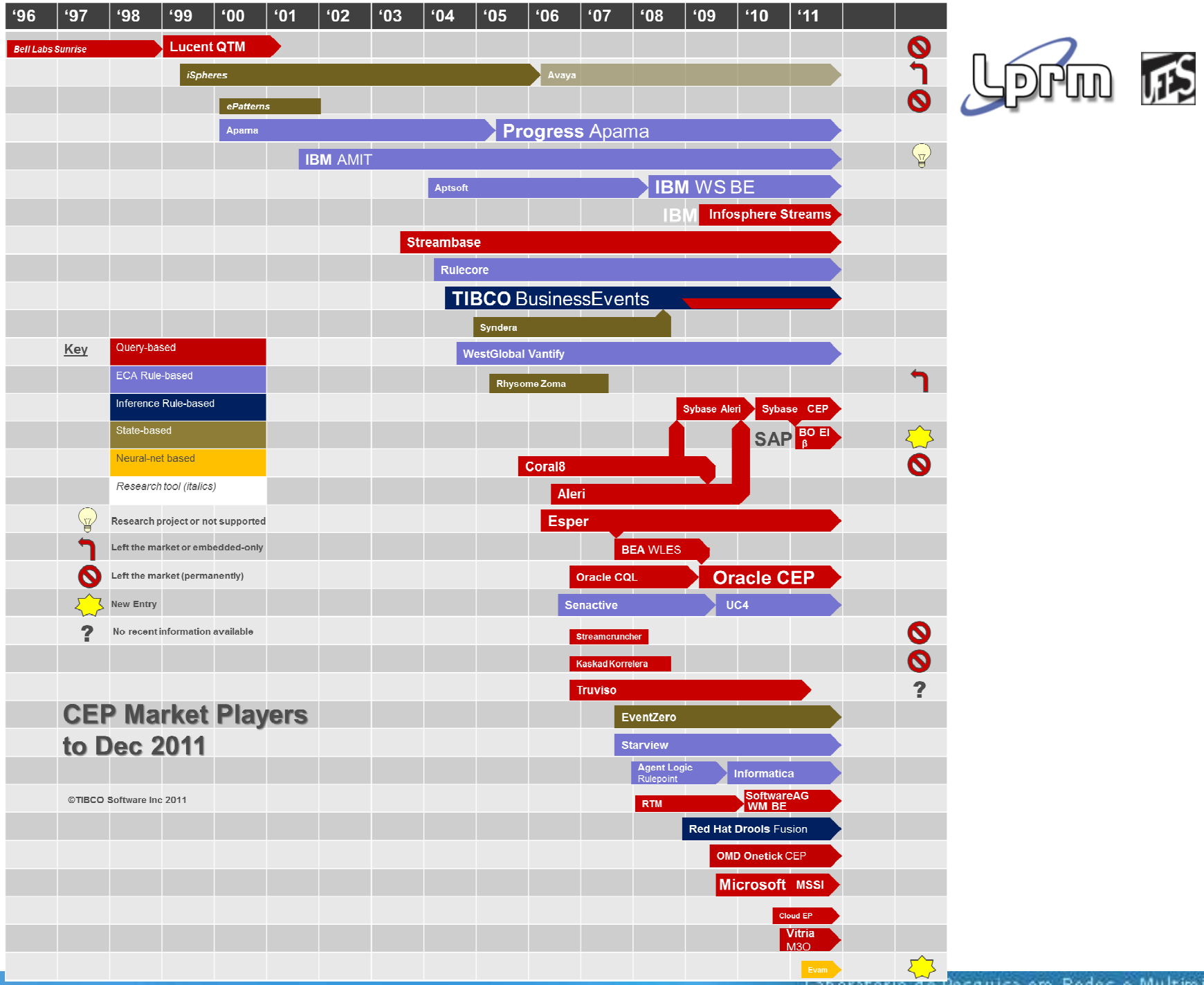
Funcionalidades CEP

Functional View source: definitions of EP



CEP – Arq. de Referencia




'96
'97
'98
'99
'00
'01
'02
'03
'04
'05
'06
'07
'08
'09
'10
'11
Bell Labs Sunrise
Lucent QTM
iSpheres
Avaya
ePatterns
Apama
Progress Apama
IBM AMIT
Aptsoft
IBM WS BE
IBM Infosphere Streams
Streambase
Rulecore
TIBCO BusinessEvents
Syndera
WestGlobal Vantify
Rhysome Zoma
Sybase Aleri
Sybase CEP
SAP BO EI β
Coral8
Aleri
Esper
BEA WLES
Oracle CQL
Oracle CEP
Senactive
UC4
Streamcruncher
Kaskad Korrelera
Truviso
EventZero
Starview
Agent Logic Rulepoint
Informatica
RTM
SoftwareAG WM BE
Red Hat Drools Fusion
OMD Onetick CEP
Microsoft MSSI
Cloud EP
Vitria M3O
Evam
Key
Query-based
ECA Rule-based
Inference Rule-based
State-based
Neural-net based
Research tool (italics)
Research project or not supported
Left the market or embedded-only
Left the market (permanently)
New Entry
No recent information available
CEP Market Players to Dec 2011
©TIBCO Software Inc 2011

Linguagens para processamento de eventos

- Linguagens de processamento de “stream” de eventos
- Linguagens orientadas a regras
- Linguagens orientadas a agentes
- Etc.

Parâmetros para descrevermos as linguagens

- Modelo de Eventos (dados e meta-dados)
- Estado dos Eventos (passados, atuais, etc)
- Modelo de Execução
- Modelo de Programação

Modelo de Eventos

- Estruturas dos eventos
- Tipos de dados para “payload” de eventos
- Identificadores de eventos
- Timestamps e ordenação
- Relações entre eventos

Estado dos Eventos

- Permite gerenciar explicitamente o estado (ciclo de vida) dos eventos?
  - Mudança de estado
  - Duração
  - Etc.

Modelo de Execução

- Execução imediata ou “atrasada”?
- Garante processamento em tempo real?
- Semântica de “time point” e “time interval”?
- Fluxo explícito de eventos?

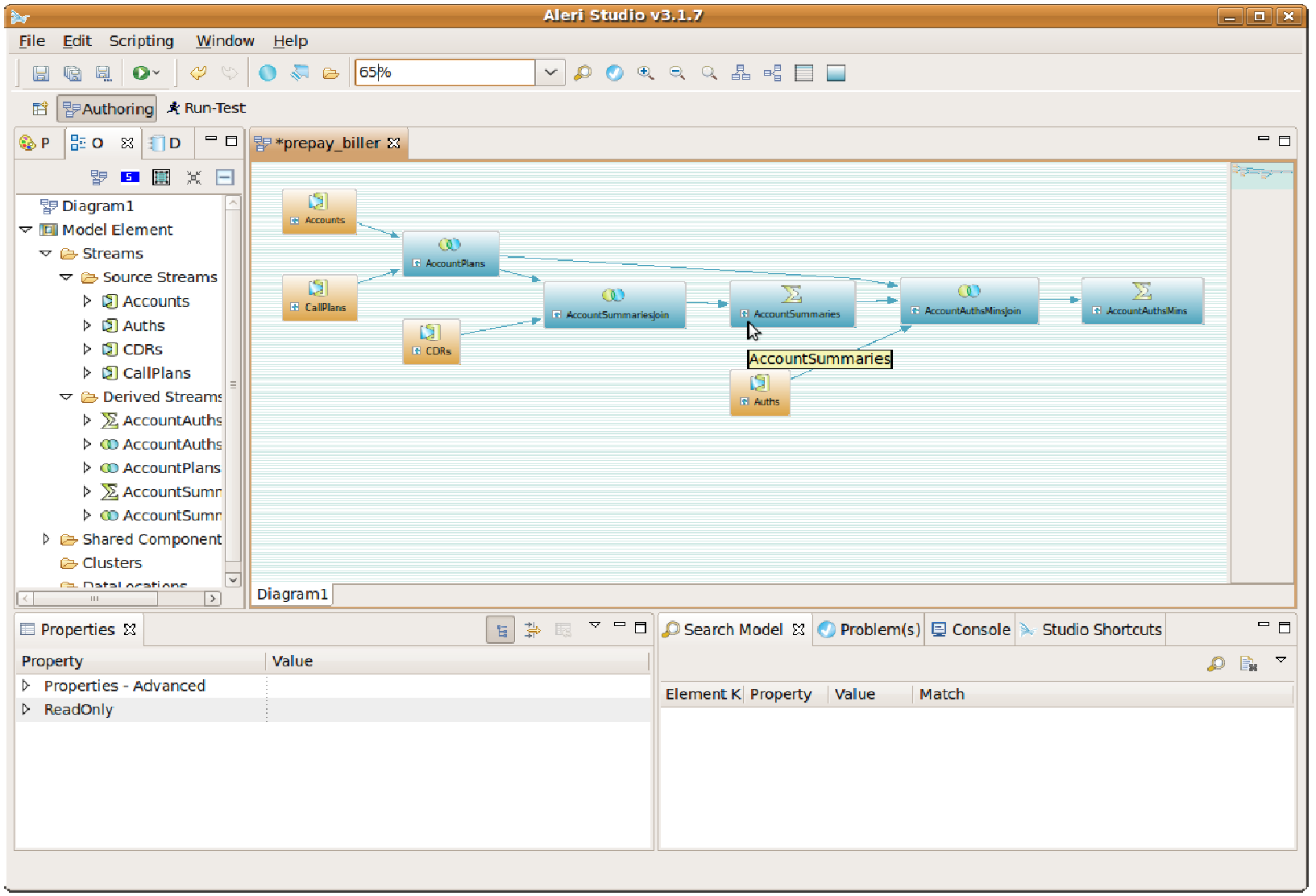
Modelo de Programação

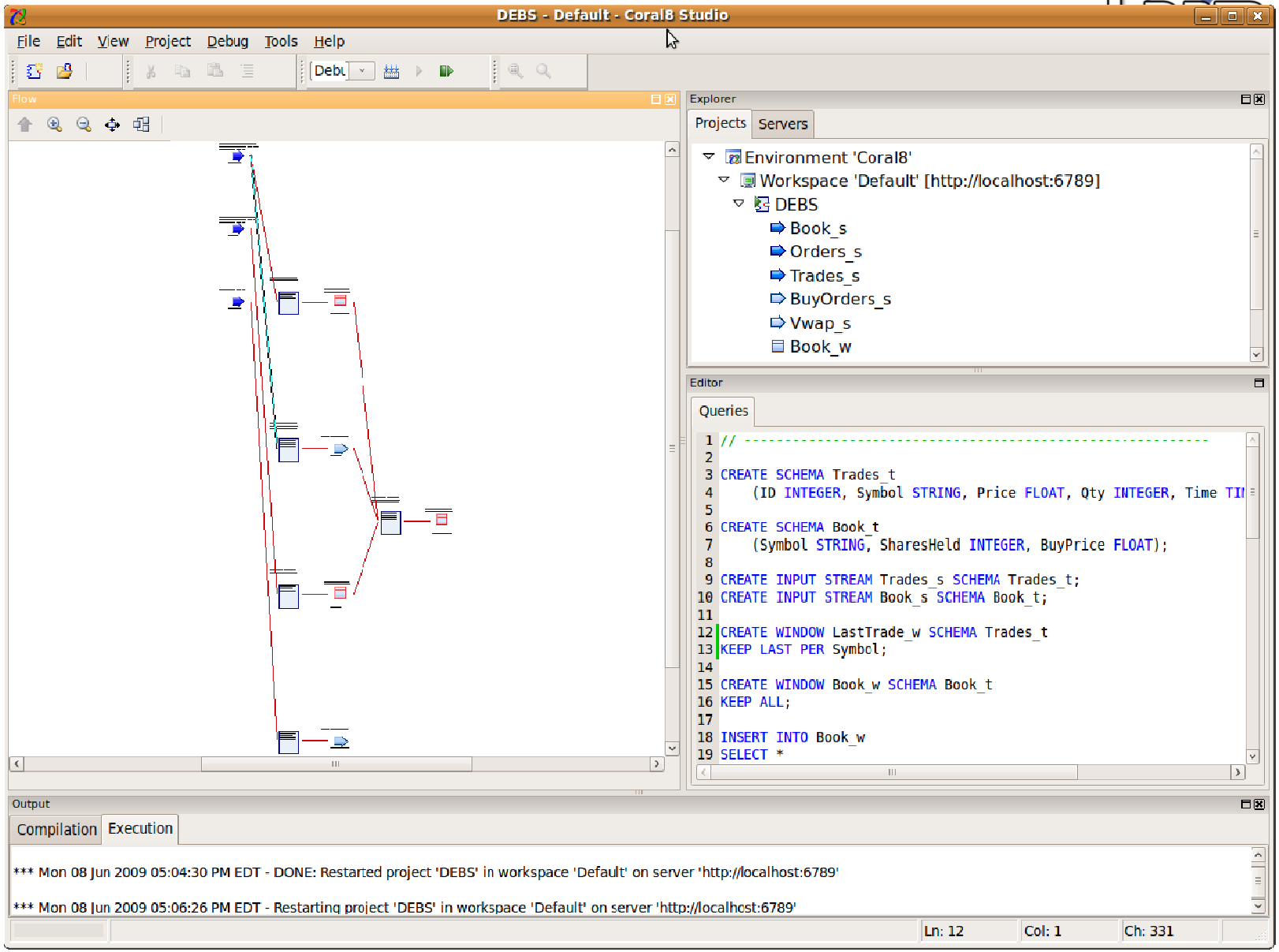
- Estilo (rule-based, stream-based, etc.)
- Abstrações
- Casamento de padrões
- “Enrichment capabilities”
- Composição
- Filtragem
- Etc.

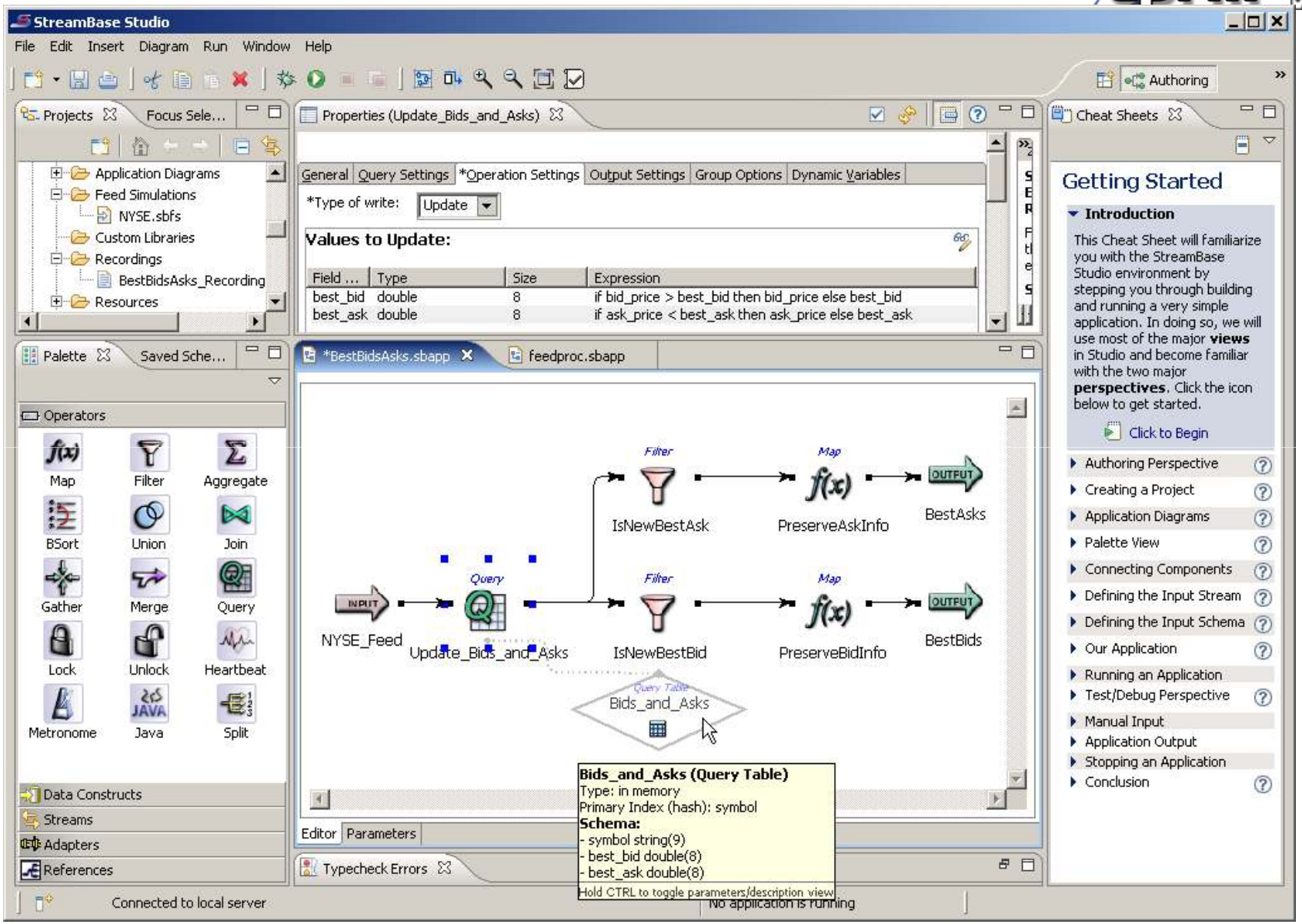
“Stream Processing”

- Eventos são processados em um grafo direcionado
  - Nós: Elementos que executam computação
  - Arestas: Comunicação de eventos entre nós
- Linguagens são inspiradas por algebra relacional (SQL-like)
- Diferença: Primeiro escrevemos a query e o resultado é atualizado na medida que os dados chegam
- Exemplos da academia: STREAM, Aurora, Cayuga, …
- Exemplos comerciais: Aleri, Coral8, Esper/Nesper, StreamBase, System S, …

Exemplos...


Aleri Studio v3.1.7
File Edit Scripting Window Help
65%
Authoring
Run-Test
*prepay_biller
Diagram1
Model Element
Streams
Source Streams
Accounts
Auths
CDRs
CallPlans
Derived Streams
Shared Components
Clusters
Accounts
AccountPlans
CallPlans
CDRs
AccountSummariesJoin
AccountSummaries
AccountAuthsMinsJoin
AccountAuthsMins
Auths
AccountSummaries
Diagram1
Properties
Property
Value
Properties - Advanced
ReadOnly
Search Model
Problem(s)
Console
Studio Shortcuts
Element K
Property
Value
Match


DEBS - Default - Coral8 Studio
File Edit View Project Debug Tools Help
Flow
Explorer
Projects Servers
Environment 'Coral8'
Workspace 'Default' [http://localhost:6789]
DEBS
Book_s
Orders_s
Trades_s
BuyOrders_s
Vwap_s
Book_w
Editor
Queries
1 // ----------
2
3 CREATE SCHEMA Trades_t
4 (ID INTEGER, Symbol STRING, Price FLOAT, Qty INTEGER, Time TI
5
6 CREATE SCHEMA Book_t
7 (Symbol STRING, SharesHeld INTEGER, BuyPrice FLOAT);
8
9 CREATE INPUT STREAM Trades_s SCHEMA Trades_t;
10 CREATE INPUT STREAM Book_s SCHEMA Book_t;
11
12 CREATE WINDOW LastTrade_w SCHEMA Trades_t
13 KEEP LAST PER Symbol;
14
15 CREATE WINDOW Book_w SCHEMA Book_t
16 KEEP ALL;
17
18 INSERT INTO Book_w
19 SELECT *
Output
Compilation Execution
*** Mon 08 Jun 2009 05:04:30 PM EDT - DONE: Restarted project 'DEBS' in workspace 'Default' on server 'http://localhost:6789'
*** Mon 08 Jun 2009 05:06:26 PM EDT - Restarting project 'DEBS' in workspace 'Default' on server 'http://localhost:6789'
Ln: 12
Col: 1
Ch: 331


StreamBase Studio
File Edit Insert Diagram Run Window Help
Authoring
Projects
Focus Sele...
Application Diagrams
Feed Simulations
NYSE.sbfs
Custom Libraries
Recordings
BestBidsAsks_Recording
Resources
Properties (Update_Bids_and_Asks)
General
Query Settings
*Operation Settings
Output Settings
Group Options
Dynamic Variables
*Type of write:
Update
Values to Update:
Field ...
Type
Size
Expression
best_bid double 8 if bid_price > best_bid then bid_price else best_bid
best_ask double 8 if ask_price < best_ask then ask_price else best_ask
Cheat Sheets
Getting Started
Introduction
This Cheat Sheet will familiarize you with the StreamBase Studio environment by stepping you through building and running a very simple application. In doing so, we will use most of the major views in Studio and become familiar with the two major perspectives. Click the icon below to get started.
Click to Begin
Authoring Perspective
Creating a Project
Application Diagrams
Palette View
Connecting Components
Defining the Input Stream
Defining the Input Schema
Our Application
Running an Application
Test/Debug Perspective
Manual Input
Application Output
Stopping an Application
Conclusion
Palette
Saved Sche...
Operators
Map
Filter
Aggregate
BSort
Union
Join
Gather
Merge
Query
Lock
Unlock
Heartbeat
Metronome
Java
Split
Data Constructs
Streams
Adapters
References
*BestBidsAsks.sbapp
feedproc.sbapp
Filter
Map
IsNewBestAsk
PreserveAskInfo
BestAsks
Query
Filter
Map
INPUT
NYSE_Feed
Update_Bids_and_Asks
IsNewBestBid
PreserveBidInfo
OUTPUT
BestBids
Query Table
Bids_and_Asks
Bids_and_Asks (Query Table)
Type: in memory
Primary Index (hash): symbol
Schema:
- symbol string(9)
- best_bid double(8)
- best_ask double(8)
Hold CTRL to toggle parameters/description view
Editor
Parameters
Typecheck Errors
Connected to local server
No application is running

Base Histórica

- Banco de dados relacionais
- Processamento de sinais

Características comuns em Stream Processing

- Eventos são registros imutáveis (campos com valores)
  - Valores tem tipos definidos (integer, float, string, blob, xml, etc.)
- Metadados: descrevem as streams e tipos de registros

Características comuns em Stream Processing- cont.

- Estado: Eventos processados e não processados
- Modelo Computacional: registros caminham no fluxo do grafo
- Modelo de Programação: Extensão de SQL/álgebra relacional

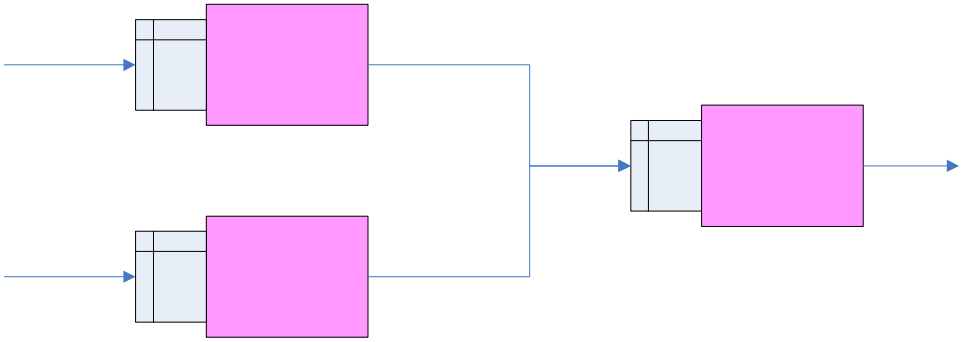
Variações...

- Eventos: diferenças no tipos
- Estado
  - Persistencia em disco? O que é persistido?
- Modelo de Programação
  - Visual ou text-based?
  - Embarcado em alguma plataforma?

Variações...

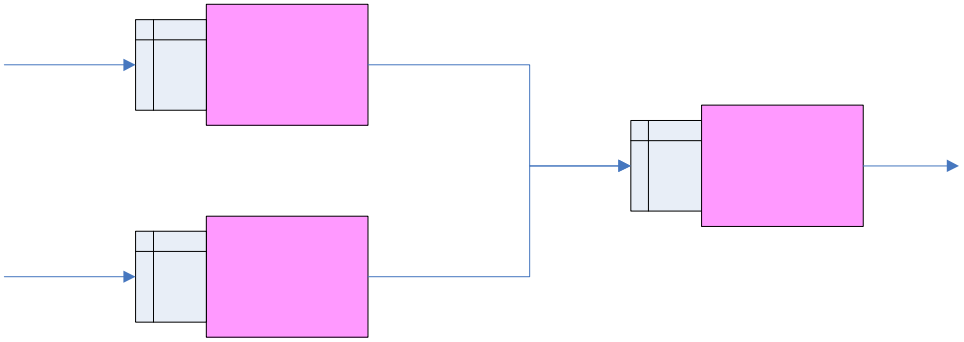
- Modelo Computacional
  - Permite ciclos no grafo?
  - Determinístico?
  - Processamento paralelo ou concorrente?
  - Computação distribuída?
  - Comunicação síncrona ou assíncrona?

Aleri Streaming Platform Dataflow Networks



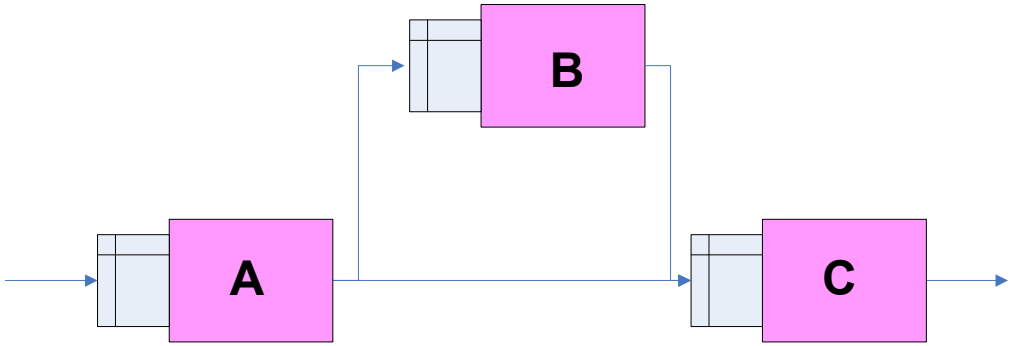
- One queue per node (instead of per edge)
- Nodes store records in table
- Events (insert/update/delete) change the table
- Nodes run in separate threads
- Optional persistence with roll-forward recovery

Aleri Streaming Platform Dataflow Networks



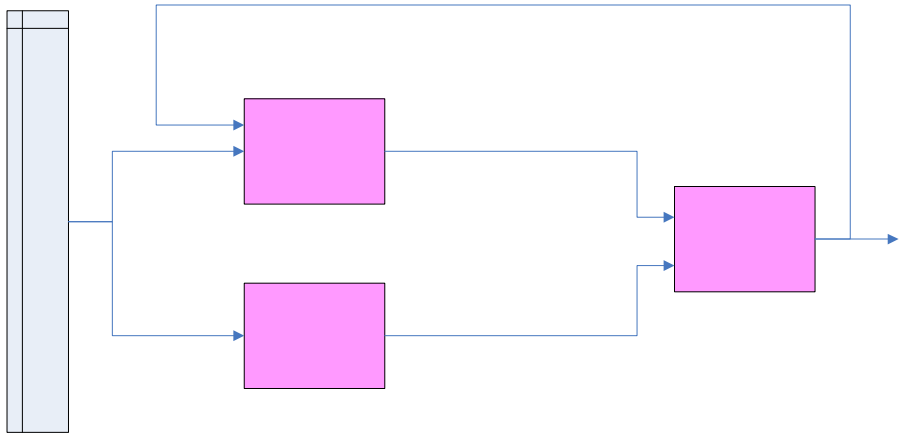
- **Computational model**
  - Events may arrive in different orders, and may cause different output events
  - Correctness: standard operators (join, compute, …) satisfy “eventual consistency”: the sequence of insert/update/deletes will produce the same table, no matter how the events flow through the graph

Aleri Streaming Platform Dataflow Networks



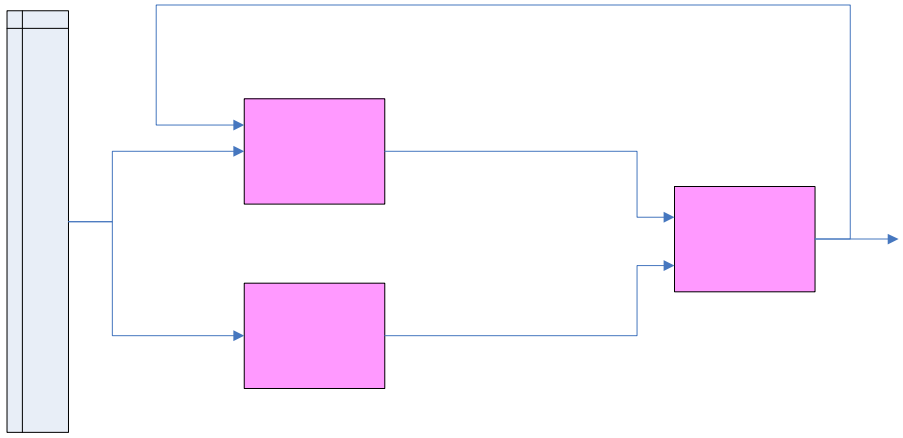
- Node A sends insert:[Id=3, Symbol=IBM, Price=45.0] Node B sends update:[Symbol=IBM, AvgPrice=43.40]
- Different arrival orders can cause different events
  - (A then B)
  - insert: [Id=3, Symbol=IBM, Price=45.0, AvgPrice=42.00]
  - update: [Id=3, Symbol=IBM, Price=45.0, AvgPrice=43.40]
  - (B then A)
  - insert: [Id=3, Symbol=IBM, Price=45.0, AvgPrice=43.40]

Coral8 Dataflow Networks



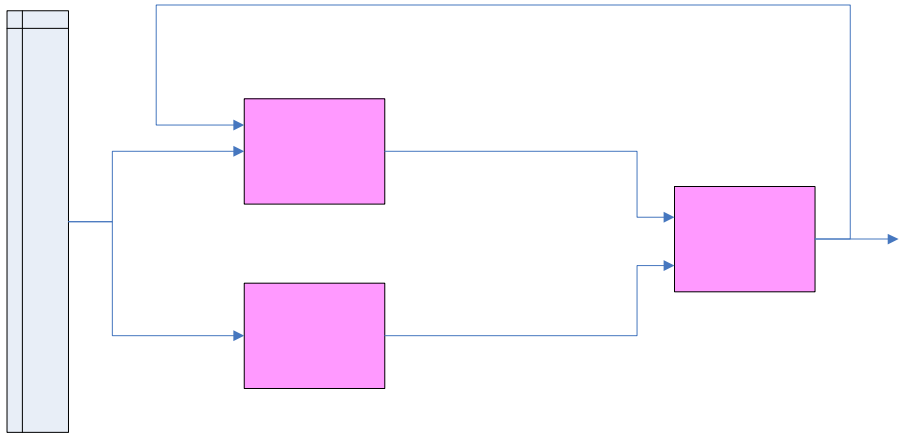
- **No queues; synchronization buffer in front**
- **Two kinds of nodes:**
  - Stream nodes process records but don't store them
  - Window nodes process and store records
- **Optional persistence for unprocessed events, records in windows, aggregate state**

Coral8 Dataflow Networks



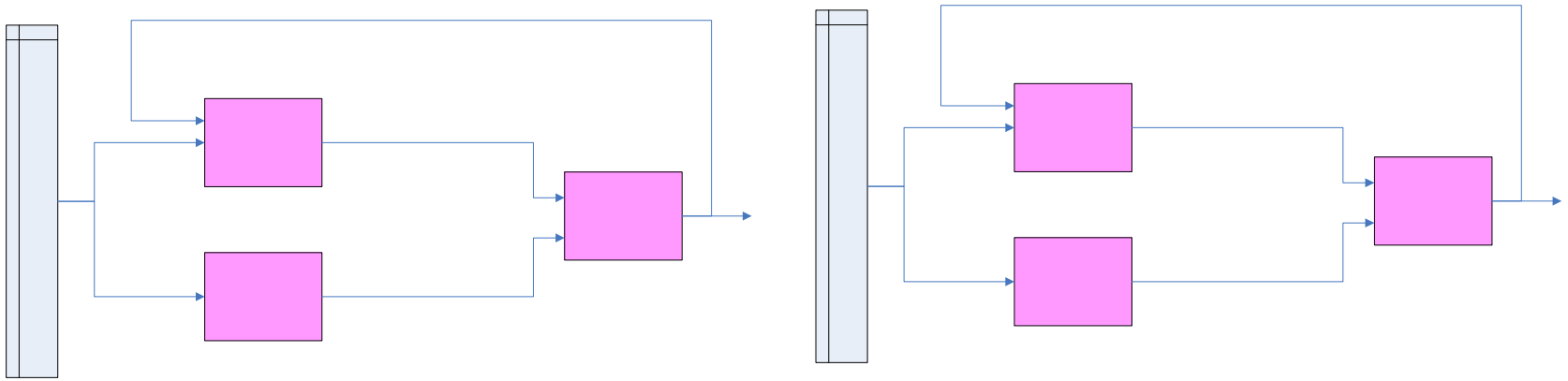
- **Computational model (simplified)**
  - External data flows into synchronization buffer
  - Events carry a timestamp (point in time, microsecond granularity): set on arrival, or by source
  - Events with same timestamp are fed into the network in one step, and are processed until network quiesces

Coral8 Dataflow Networks



- **Computational model**
  - Deterministic: CCL programs produce the same output events given the same input
  - Priorities assigned to nodes in graph to maintain determinacy

Coral8 Dataflow Networks



- **Computational model: concurrent/distributed**
  - Project = group of CCL streams, windows, queries (code for calculating new events from old)
  - Projects can send/receive events to/from other projects
  - Each project has a thread (there are other threads too)
  - Projects can be distributed across machines

Examples in CCL: Simple Market Data

- Schema: type of records
  - `CREATE SCHEMA Trades_t`
  - `(Symbol STRING, Qty INTEGER, Price FLOAT);`
  - `CREATE SCHEMA Book_t`
  - `(Symbol STRING, Qty INTEGER, BuyPrice FLOAT);`
- Other scalar types: BOOLEAN, TIMESTAMP, INTERVAL, XML, BLOB
- CCL streams: INPUT, OUTPUT, local
  - `CREATE INPUT STREAM Trades_s SCHEMA Trades_t;`
  - `CREATE INPUT STREAM Book_s SCHEMA Book_t;`

CCL Windows

- Windows are like streams, but store records
  - `CREATE WINDOW Book_w SCHEMA Book_t KEEP ALL;`
  - `INSERT INTO Book_w`
  - `  SELECT * FROM Book_s;`
- Sample of KEEP policies (there are others):
  - `KEEP LAST PER Id`
  - `KEEP 3 MINUTES`
  - `KEEP EVERY 3 MINUTES`
  - `KEEP UNTIL (”MON 17:00:00”)`
  - `KEEP 10 ROWS`
  - `KEEP LAST ROW`
  - `KEEP 10 ROWS PER Symbol`

CCL Queries: SQL-based syntax

- Aggregation
  - `CREATE STREAM Vwap_s SCHEMA (Symbol STRING, Vwap FLOAT);`
  - `INSERT INTO Vwap_s`
  - `  SELECT Symbol, sum(Qty * Price)/sum(Qty)`
  - `  FROM Trades_s KEEP 30 MINUTES`
  - `  GROUP BY Symbol;`
- Join
  - `CREATE SCHEMA Value_t`
  - `  INHERITS FROM Book_t (CurVal FLOAT, AvgVal FLOAT);`
  - `CREATE WINDOW BookValue_w`
  - `  SCHEMA Value_t KEEP LAST PER Symbol;`
  - `INSERT INTO BookValue_w`
  - `  SELECT B.Symbol, B.SharesHeld, B.BuyPrice,`
  - `  B.SharesHeld * L.Price, B.SharesHeld*V.Vwap`
  - `  FROM Book_w B, LastTrade_w L, Vwap_s V`
  - `  WHERE B.Symbol = L.Symbol and B.Symbol = V.Symbol;`

CCL Pattern Matching

- MATCHING clause
  - `INSERT INTO OrdersMatch_s`
  - `  SELECT B.ID, S.ID, B.Price`
  - `  FROM BuyOrders_s B, Trades_s T, SellOrders_s S,`
  - `  MATCHING [30 SECONDS: B, !T, S]`
  - `  ON B.Price = S.Price = T.Price;`
- Operators in patterns (cf. finite-state automata):
  - Three boolean operators: ! (not), & (and), | (or)
  - One temporal operator: , (followed by)
  - Temporal scope can be nested
    - `  MATCHING [30 SECONDS: [10 SECONDS: B, !T], S]`

Características de outras linguagens

- Esper (Java)/Nesper (C#)
  - SQL- like
  - Embarcada na linguagem host
- AleriML
  - SPLASH: linguagem C-like
  - Estruturas de dados : vetores, dicionários, event caches
- StreamBase
  - LOCK/UNLOCK para controlar fluxo de registros

Regras de Produção

Regras de Produção e CEP

- Regras de produção reagem a mudança de estado (e não eventos)
- Sistemas de RP com modelo de objeto e atualizações externas de fatos podem ser estendidos para apoiar CEP
  - Tipos de Eventos e classes são definidos na declaração das regras
  - Novas instancias de eventos são inseridos na working memory
  - As instâncias de eventos são filtrados (dos fluxos) e combinados através de casamento de padrões e se as condições forem satisfeitas, a ação é invocada
- Podem ser estendidos com mecanismos de query, condições temporais, etc.
- Exemplos: TIBCO Business Events, Drools

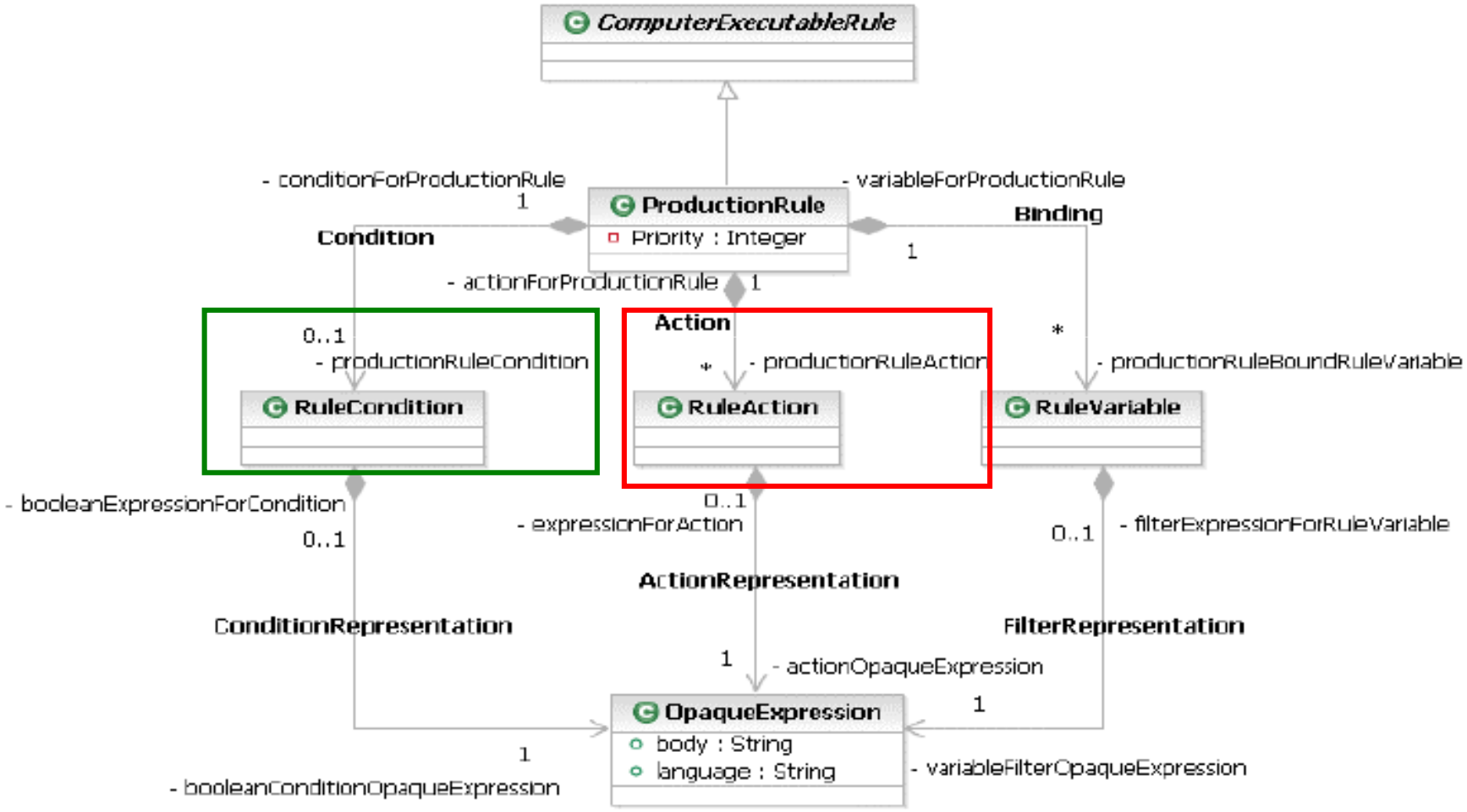
Regras de Produção : Abordagem da OMG (OMG PRR)

- Representação “neutra” em UML para regras de produção
  - Fair Isaac, IBM/ILOG, LibRT, Pega, Corticon, TIBCO
  - RuleML Group
  - Fujitsu

OMG Production Rules Representation

- Modelo PIM (MDA)
  - Estende UML de forma a incluir regras como "1st class citizens" do modelo
  - Não inclui uma linguagem específica
- Pode utilizar outros padrões como linguagem
  - e.g. W3C Rule Interchange Format (W3C RIF) e RuleML

OMG PRR ProductionRule Classes



if [condition] then [action]

W3C Rule Interchange Format

- W3C Rule Interchange Format (W3C RIF)

  http://www.w3.org/2005/rules/wiki/RIF_Working_Group
- Objetivo: Padronização de regras na Web
- Dialetos RIF 1.0:
  - RIF Production Rules Dialect (RIF-PRD)
  - RIF Basic Logic Dialect (RIF-BLD)
    - Formato para regras em lógica (regras Horn rules)

RIF Production Rule

```
<Implies>
    <if> FORMULA </if>
    <then rif:ordered="yes">
        <Do>
          ACTION*
        </Do>
    </then>
</Implies>
```

Syntax close to PR RuleML syntax

Event Condition Action Rules

Event Condition Action Rules

- ECA Rule
  "on Event if Condition do Action";
- Parte do Evento fica separada da condição
  - e.g. *expresses customer order* (event); *check if credit card is valid* (condition)
- Origem: Banco de dados ativos (active databases)
  - extend databases with reactions, e.g. HiPac, ACCOOD, Chimera, ADL, COMPOSE, NAOS
  - Composite event algebra, e.g. SAMOS, COMPOSE, Snoop
    - `Sequence | Disjunction | Xor | Conjunction | Concurrent | Not | Any | Aperiodic | Periodic`

CEP e a Web Semântica

- Real-time Web!
  - Número crescente de recursos na web que exigem mais do que “request-response”
  - HTML-5 push data to browsers
  - Facebook Graph API: real-time updates
  - Google supports push-notifications “PubSubHubBub”

Real-time Web: Desafios

- Identificar uma infraestrutura “internet-like” que seja descentralizada, global e que seja construída com padrões amplamente aceitos

- *Event Processing Fabric*
  - “Designed to be the highway of global real-time data, and the enabler of applications for a proactive society”

Event Processing Fabric : Desafios

- Milhares, milhões de “sources” diferentes
  - ... de todos os lugares do mundo
- Filtragem, agregação, transformação e detecção de padrões
- Usar dados em tempo-real e dados históricos
- Gerenciar subscrição e localização de milhões de usuários
  - ... de uma maneira segura e anônima
  - ... de diferentes lugares e domínios administrativos
- Mandar alertas de forma “time-based”
- Utilizar os canais de comunicação mais apropriados
- Etc., etc.

Event Processing Fabric : Limitações

- Pode não ser adequada para aplicações que exijam segurança
  - Aplicações militares
- Pode não ser adequada para aplicações que exijam alto desempenho
  - Stock trading

Requisitos para Web-Scale CEP

- Formato dos eventos
  - Standard, extensible
- Linguagem para deteção de Padrões de Eventos

Requisitos para Modelo de Evento

- Point in time, time-interval
- Evento como um objeto “first-class”
- Propriedades temporais
- Hierarquia de eventos
- Relação entre eventos
- Real-time modeling (not only at design-time)
- Open/extensible (todos devem ser capazes de produzir e consumir eventos)

Propostas para a Web-scale CEP

- Para modelagem de eventos
  - RDF com URI's (*CEP linked data*)
- Linguagem de padrões de eventos
  - EP-SPARQL