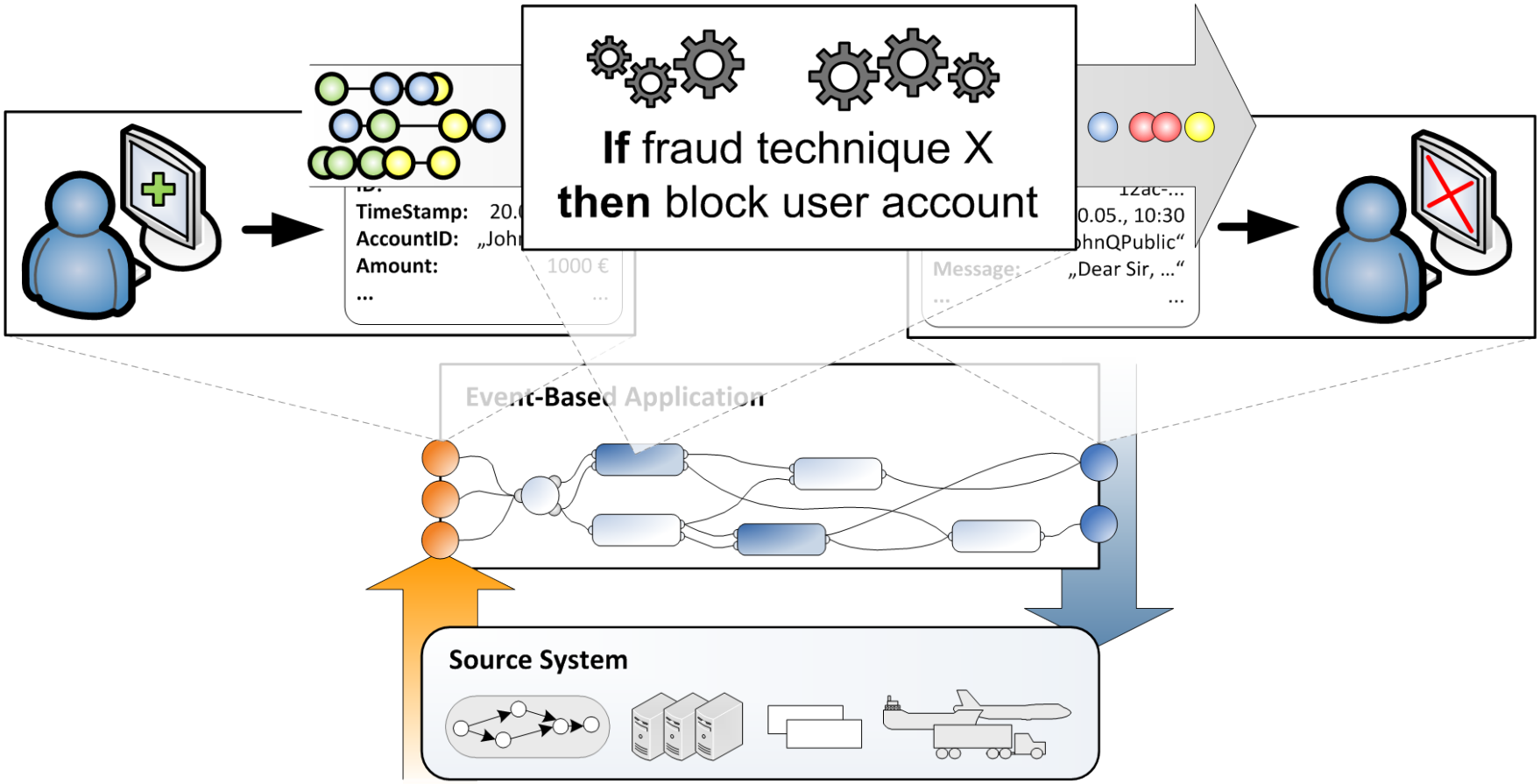
Entity-Based State Management for Complex Event Processing Applications

Hannes Obweger
UC4 Senactive
RuleML 2011 @ Europe

Rethink automation.

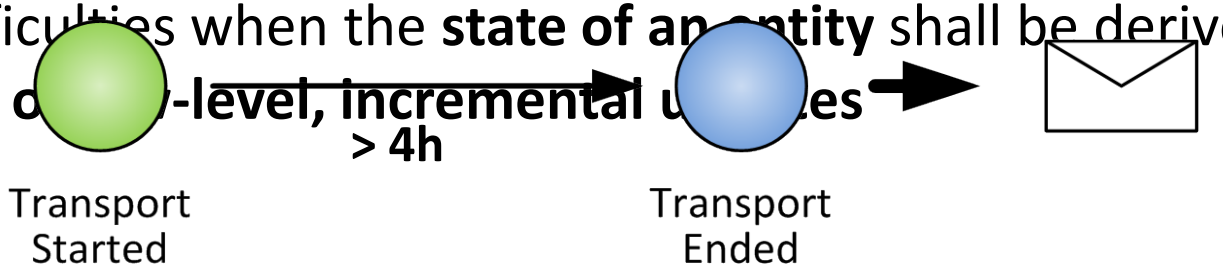
Sense-and-Respond Infrastructure

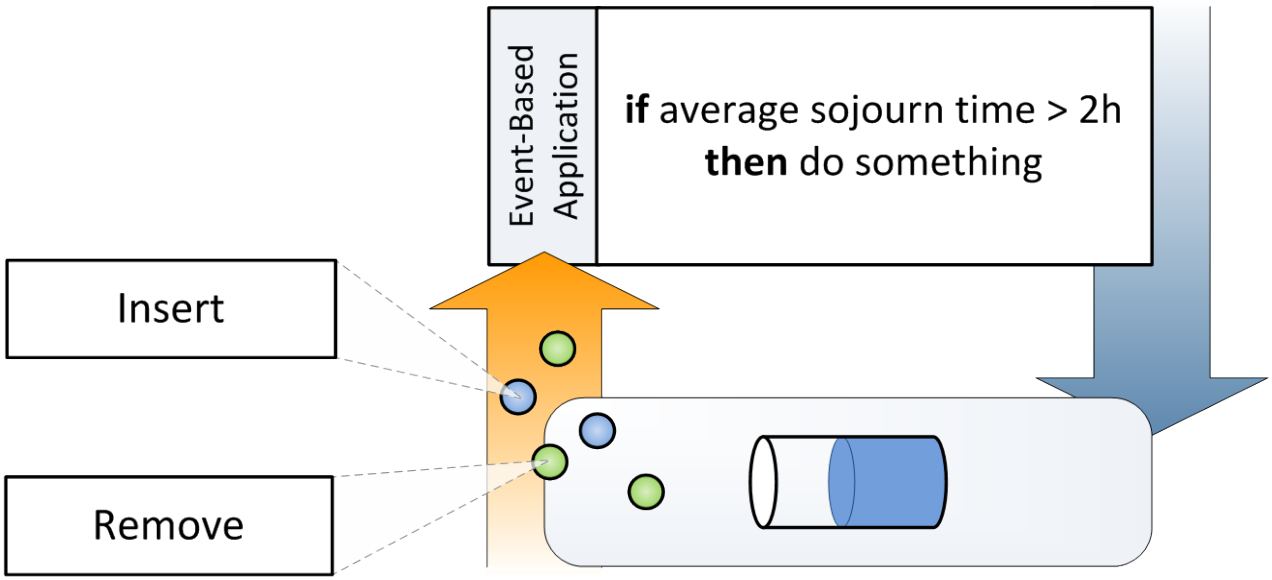


CEP using Event Condition Action (ECA) Rules

… works well for detecting situations of a **defined length and structure**

… faces difficulties when the **state of an entity** shall be derived from **a stream of low-level, incremental updates**

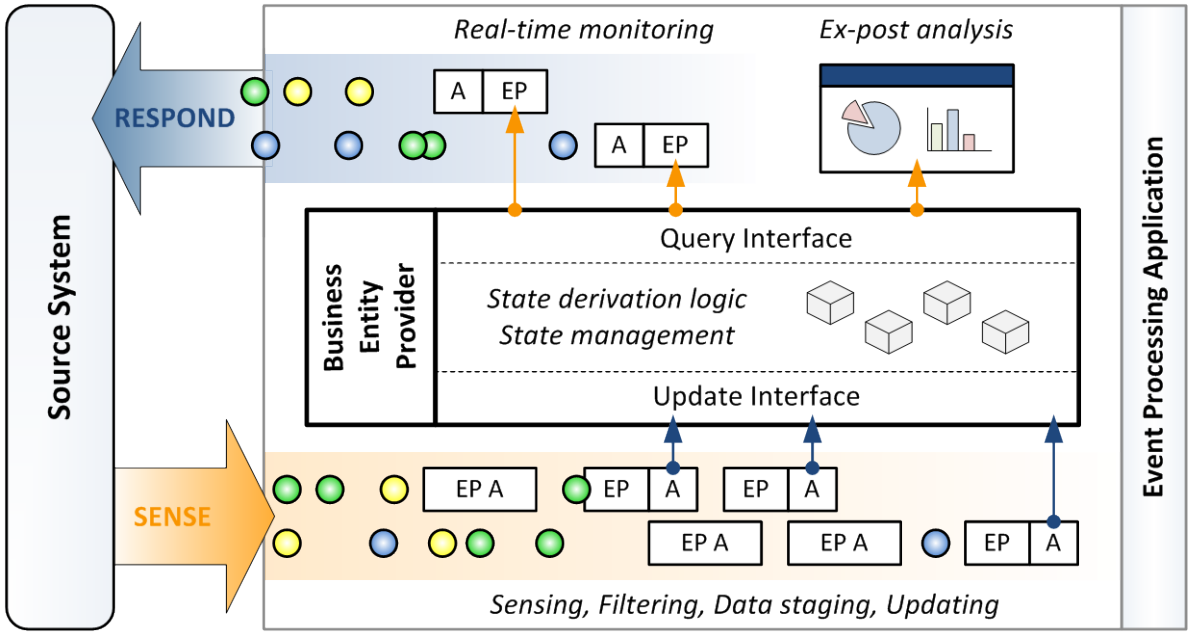




Challenges

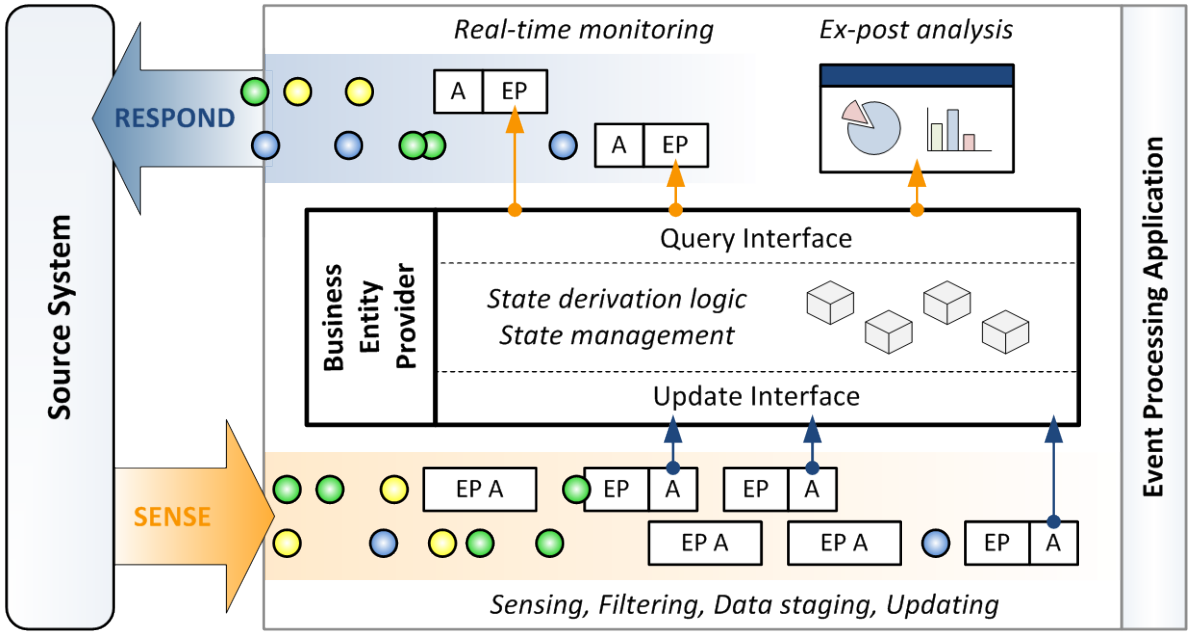
- Durable entity state
  - vs. sliding time windows
  - Specific data management for non-volatile data
  - Ex-post analysis
- Complex state calculation logic
  - Goal: Keep rules thin and understandable
- Active entity monitoring
  - vs. pull-based approach
- Context-aware data access
  - Integegration with contexts, correlation sets, …
- Ease of use
  - Accessible to business users

Business Entity Providers



- Encapsulate state calculation logic
- Manage data as system-wide data structures
- Plugged into application
- Expose easy-to-use interfaces
- Updated and queried using ECA rules

Business Entity Providers



→ Exploit CEP also in entity-centric environments

→ Simplify rule logic

→ Separation of concerns

- Low-level processing logic, high-level business logic

→ Full integration with rule model

Outline

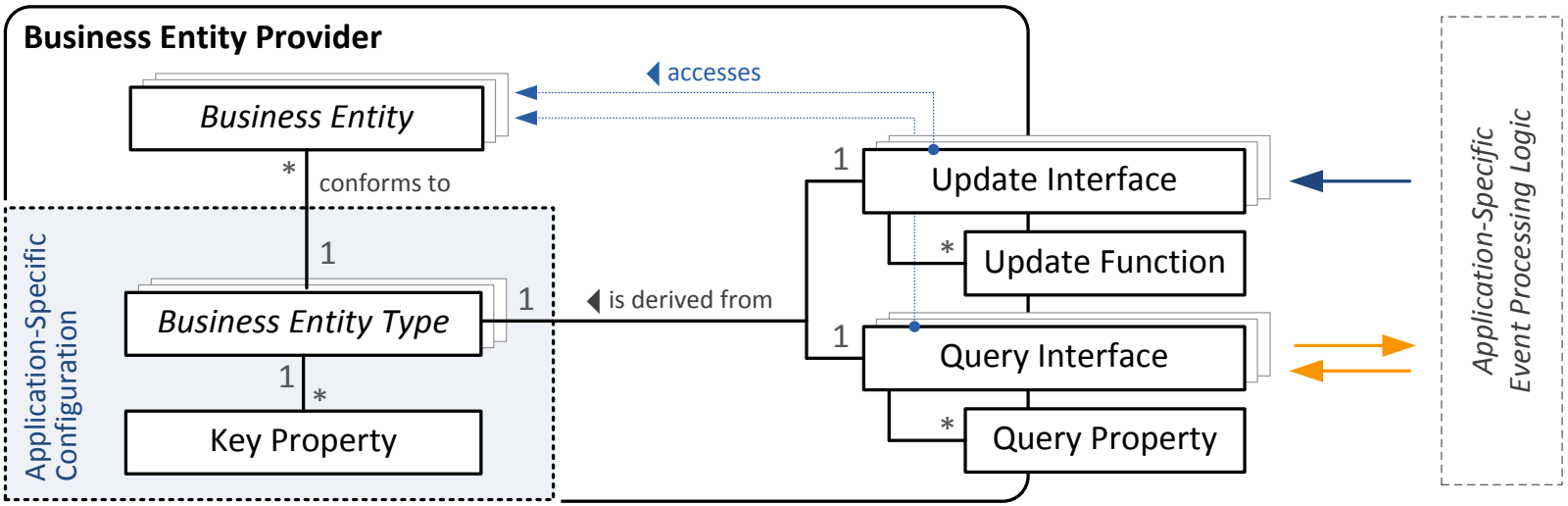
- Introduction
- **Business Entity Provider Model**
- Exemplary Business Entity Providers
- Rule Model: Basics
- Rule Model: Extensions
- Implementation
- Conclusion

Business Entity Provider Model

- Provider per „kind“ of entity
  - E.g., queues (de facto: sets)
- Generic framework: You may implement a provider in **whichever way is appropriate for your purposes**
  - E.g., RDBMS + SQL statements, Hadoop/Cassandra, Business Rule Engine, …

- **BUT:** Must adhere to a basic structure

Business Entity Provider Model



- Business Entity Type
  - E.g., Task queue
  - Key properties (→ composite key)
- Business Entity
  - E.g., Task queue #42
  - Unique, immutable key tuple
- Update and query interfaces
  - Query properties, update functions

Outline

- Introduction
- Business Entity Provider Model
- **Exemplary Business Entity Providers**
- Rule Model: Basics
- Rule Model: Extensions
- Implementation
- Conclusion

Exemplary Business Entity Providers

- Base Entity
  - Key → Set of typed entity properties
  - E.g., Customer
  - Update / Query: 'setter' and 'getter' for all entity properties
- Scores
  - Key → Numeric score value
  - E.g., Alarms per server
  - Update: set, increment, decrement
  - Query: current value, aggregates
- Sets
  - Key → Collection of set elements
  - E.g., FIFO queue, priority queue, stack
  - Update: insert, remove
  - Query: Current set of elements → calculations

Outline

- Introduction
- Business Entity Provider Model
- Exemplary Business Entity Providers
- **Rule Model: Basics**
- Rule Model: Extensions
- Implementation
- Conclusion

Rule Model: Basics

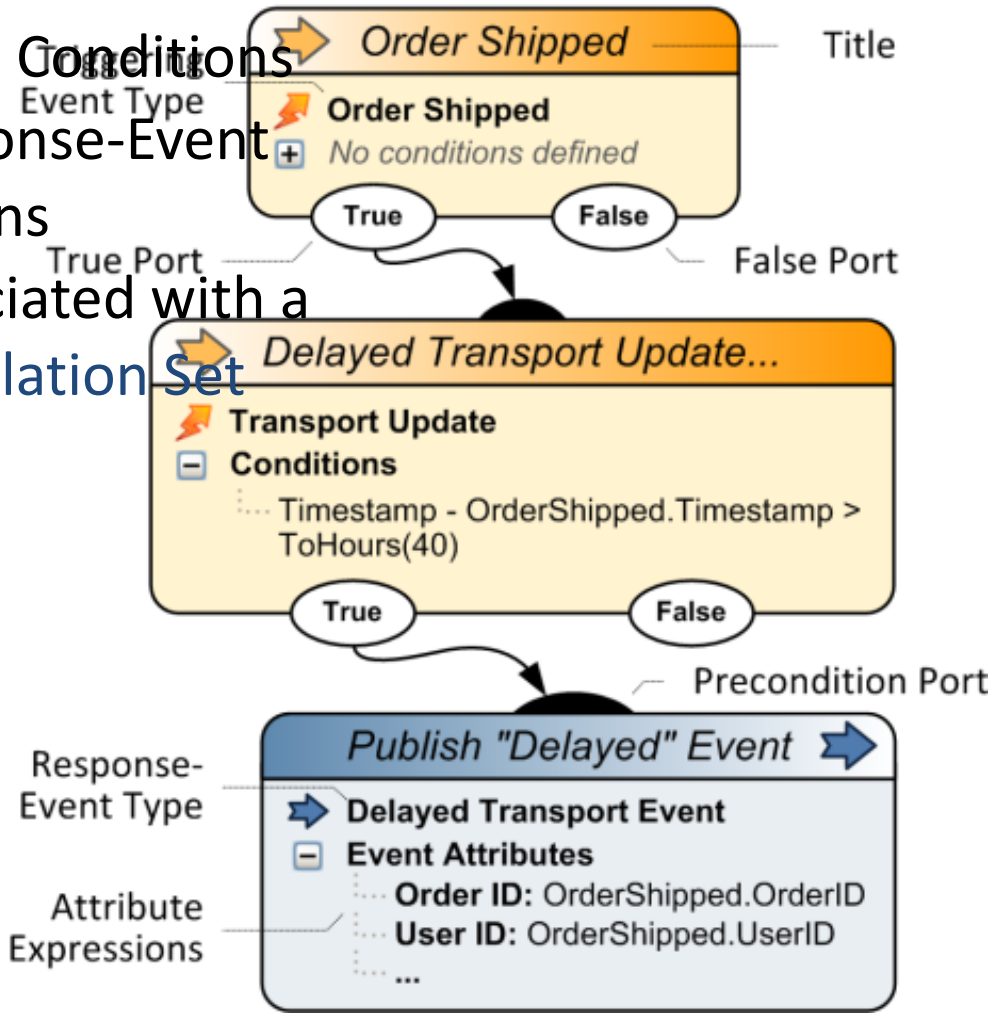
- Separation:

Event Correlation	and	Pattern Detection
Group events that are **generally related** to each other; e.g., all events of a transport process		From the resulting situations, pick those that are **noteworthy**; e.g., all transports longer than 4h

Rule Model Basics: Correlation Model

Rule Model Basics: Decision Graph Model

- Event Conditions
- Response-Event Actions
- Associated with a Correlation Set

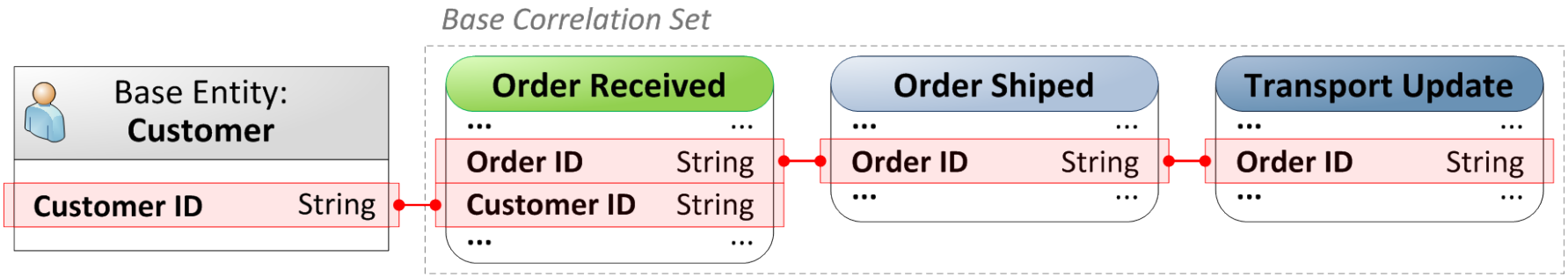


Outline

- Introduction
- Business Entity Provider Model
- Exemplary Business Entity Providers
- Rule Model: Basics
- **Rule Model: Extensions**
- Implementation
- Conclusion

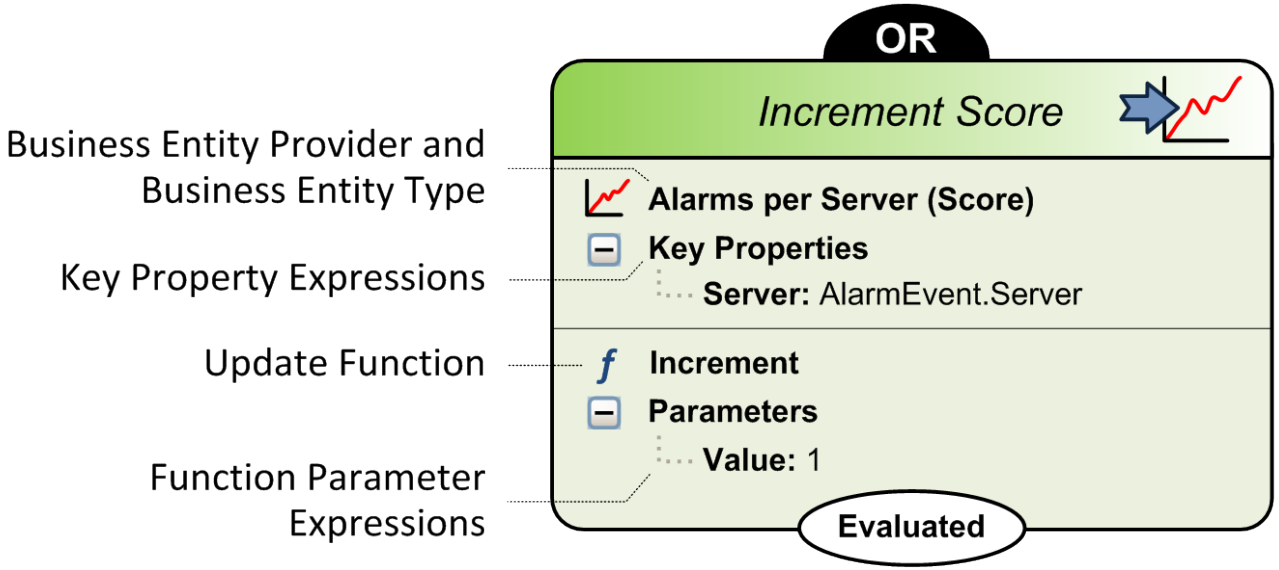
Rule Model Extensions: Correlation Model

- **Generalization:** Let users define relations between events, events and business entities, and different kinds of entities
- **Events:** Correlated based on **event attributes**
- **Business Entities:** Correlated based on **key properties**



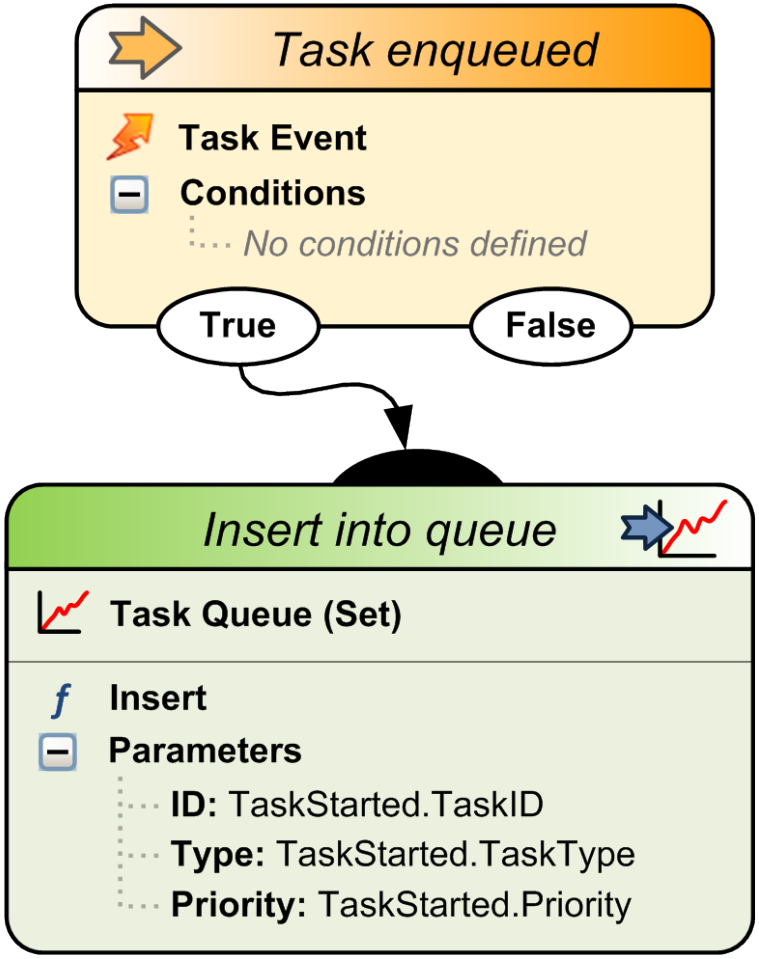
Rule Model Extensions: Business Entity Action

Whenever activated: **Update correlated business entities**



- Business entity provider
- Business entity type
- Key property expressions
- Update function (e.g., increment)
- Function parameter expressions (e.g., 1)

Rule Model Extensions: Business Entity Action

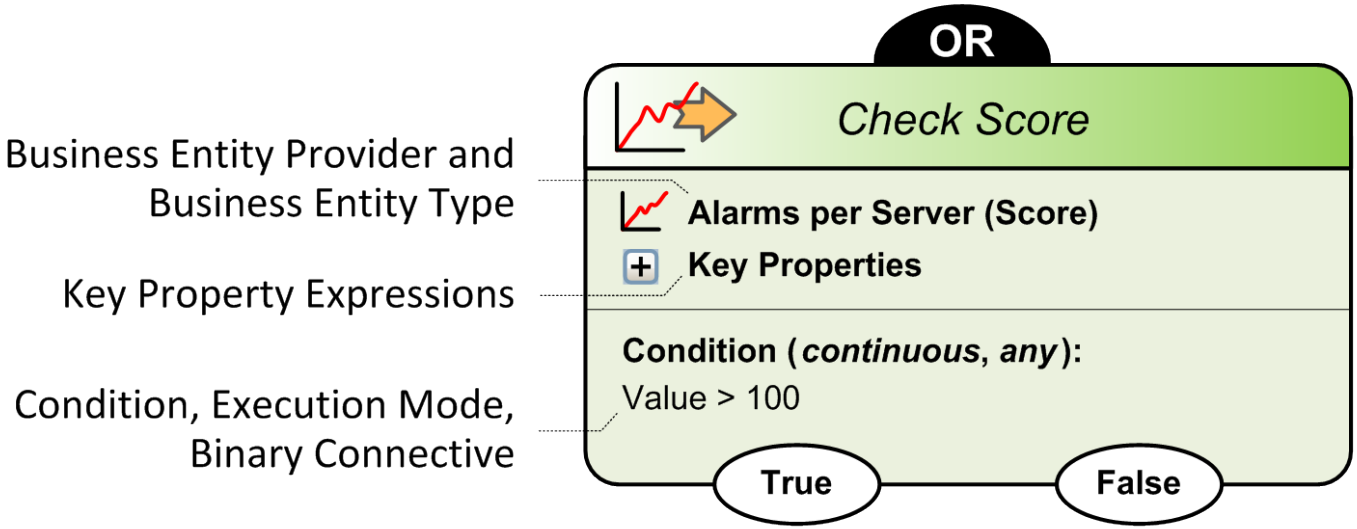


Rule Model Extensions: Business Entity Condition

Evaluate a **query** on all correlated business entities

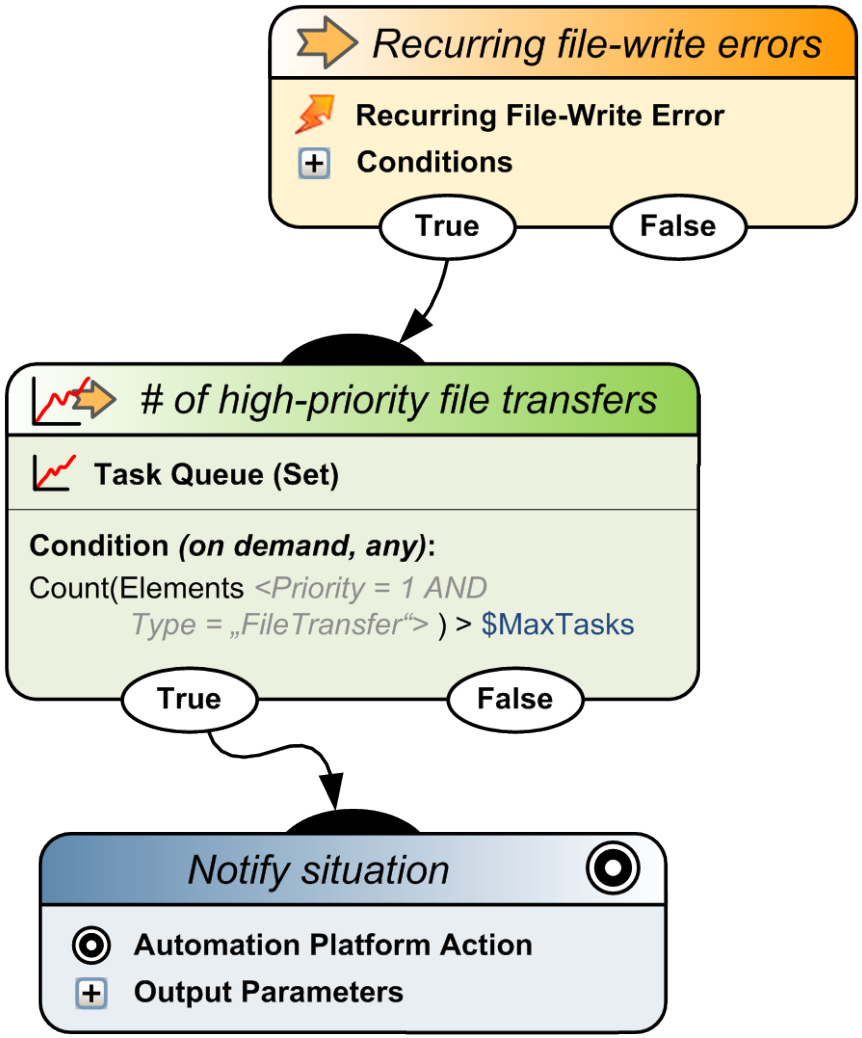
a. **whenever activated** (on-demand access)

b. **whenever an update occurs** (continuous access)

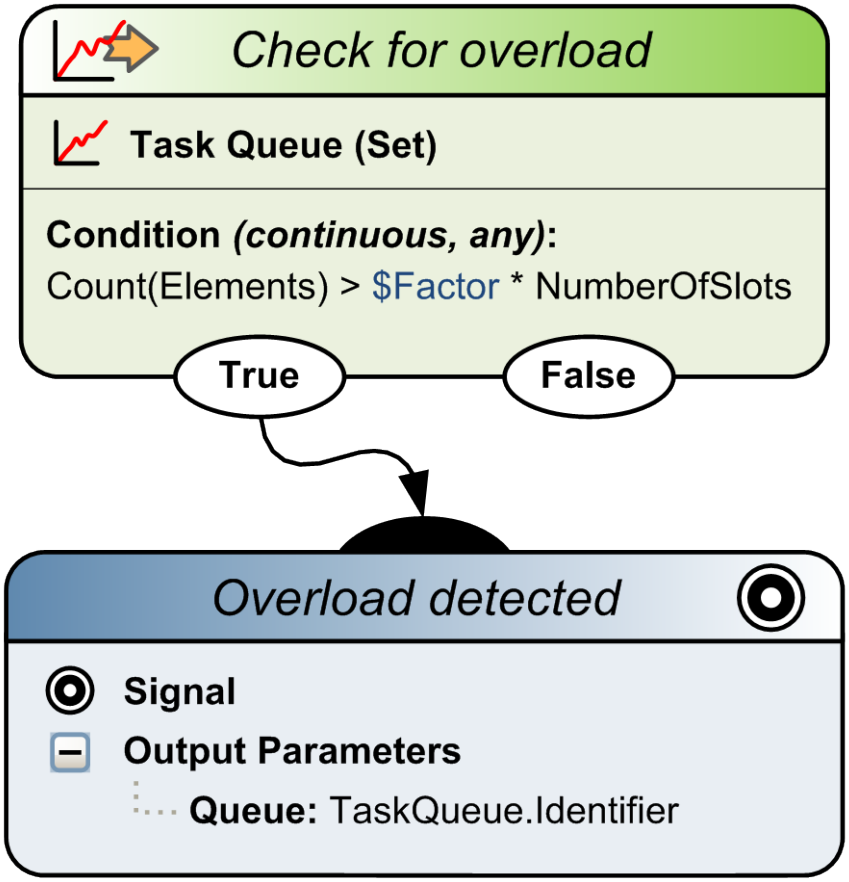


- Provider, type, key property expressions
- Boolean condition (e.g., Value > 100)
- Binary connective (all, at least one, exactly one)
- Execution mode (on demand vs. continuous)

Rule Model Extensions: Business Entity Condition



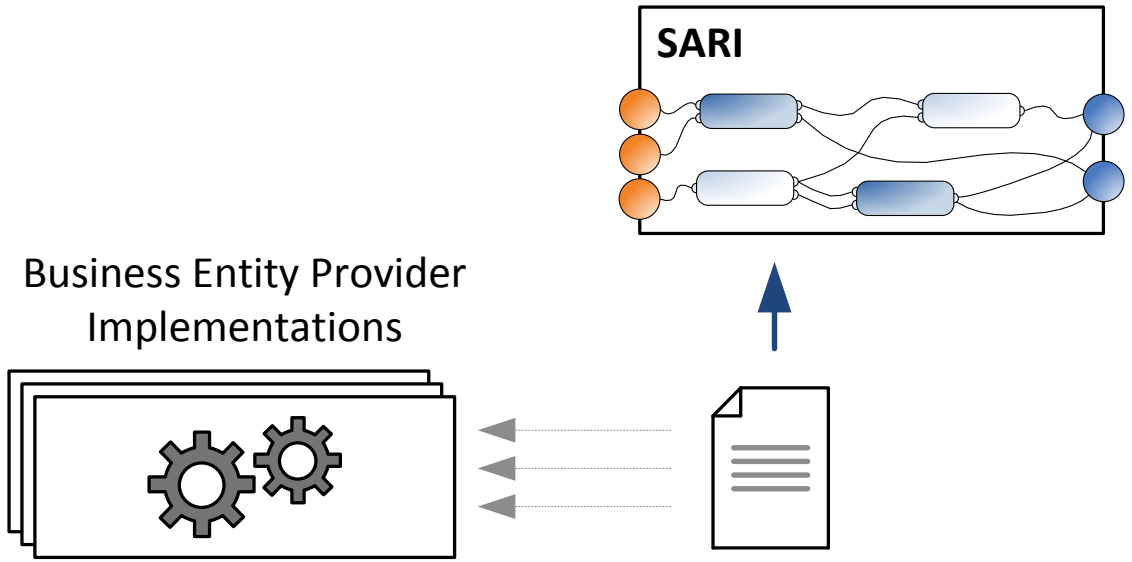
Rule Model Extensions: Business Entity Condition



Outline

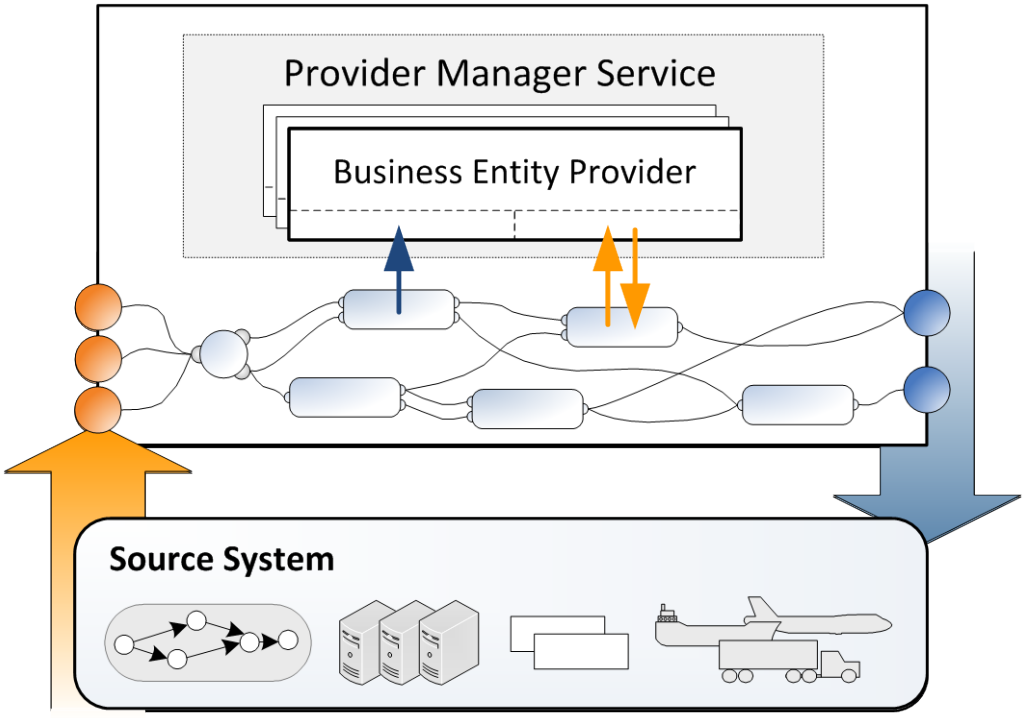
- Introduction
- Business Entity Provider Model
- Exemplary Business Entity Providers
- Rule Model: Basics
- Rule Model: Extensions
- **Implementation**
- Conclusion

Implementation: Basics



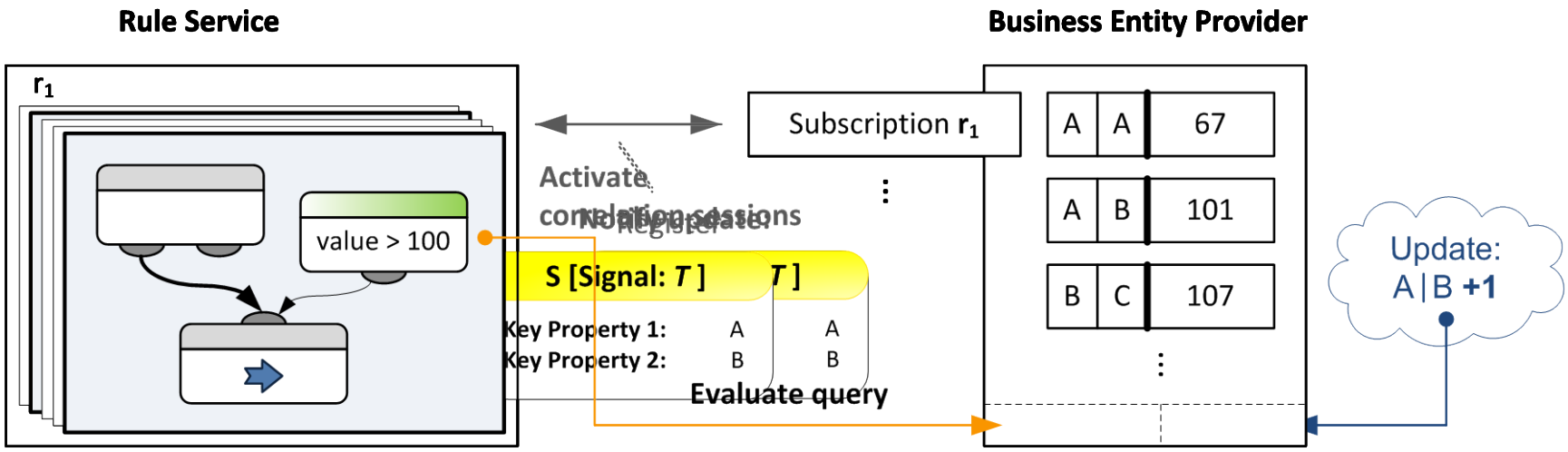
- Provided as .NET assemblies
- Referenced in an XML file
- On start up: Parsed by SARI, instantiations

Implementation: On Demand Access



- Added to globally accessible **provider management service**
- Called directly and synchronously from rule services

Implementation: Continuous Access



On start up: Register as listener at provider management

1. Update occurs
2. Special notification event is sent to all registered services
3. Use basic correlation mechanism to active decision graph
4. Evaluate key property expressions
5. Evaluate query as with on-demand access

Outline

- Introduction
- Business Entity Provider Model
- Exemplary Business Entity Providers
- Rule Model: Basics
- Rule Model: Extensions
- Implementation
- **Conclusion**

Conclusion

- Novel approach to state management for ECA-based CEP
- **Business Entity Providers**
  - Encapsulate custom state-calculation logic
  - Plugin-based implementation model
  - Fully integrate with SARI rule model

- **Durability entity state, comple state calculation** ✓
- **Active entity monitoring** ✓
- **Context-aware data access** ✓
- **Ease of use** ✓

Thanks for your attention!