ODBASE 2009

An Active Domain Node Architecture for the Semantic Web

Franz Schenk, Wolfgang May
Institut für Informatik, Universität Göttingen

Overview

I. Event-driven environment

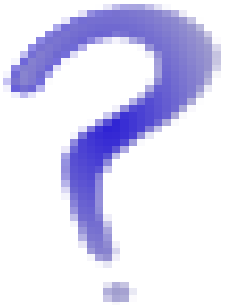
II. Domain Ontologies

III. Components of the application node

Use Case

Travel Scenario:

Book a flight from
Frankfurt to Lisboa

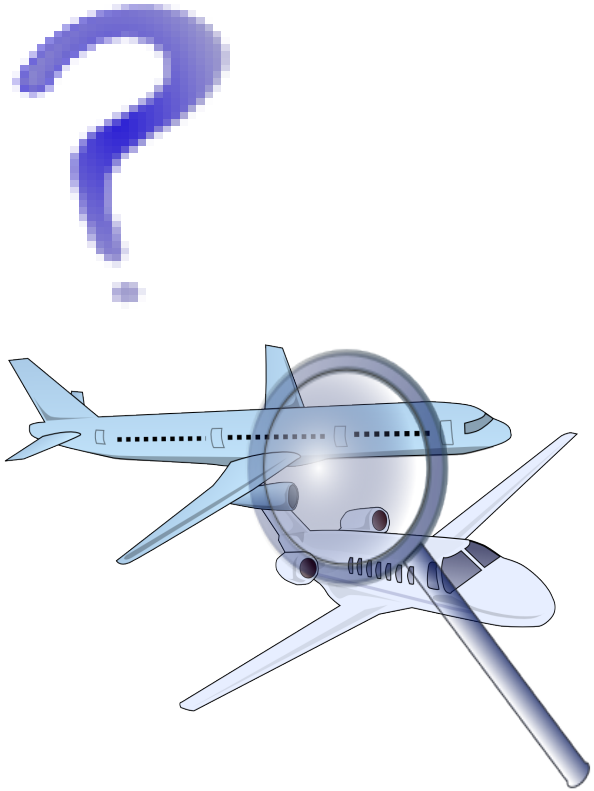


Travel Scenario:

Book a flight from
Frankfurt to Lisboa

Search for flights from FRA to LIS on November 01.

Are there available seats for that flight?



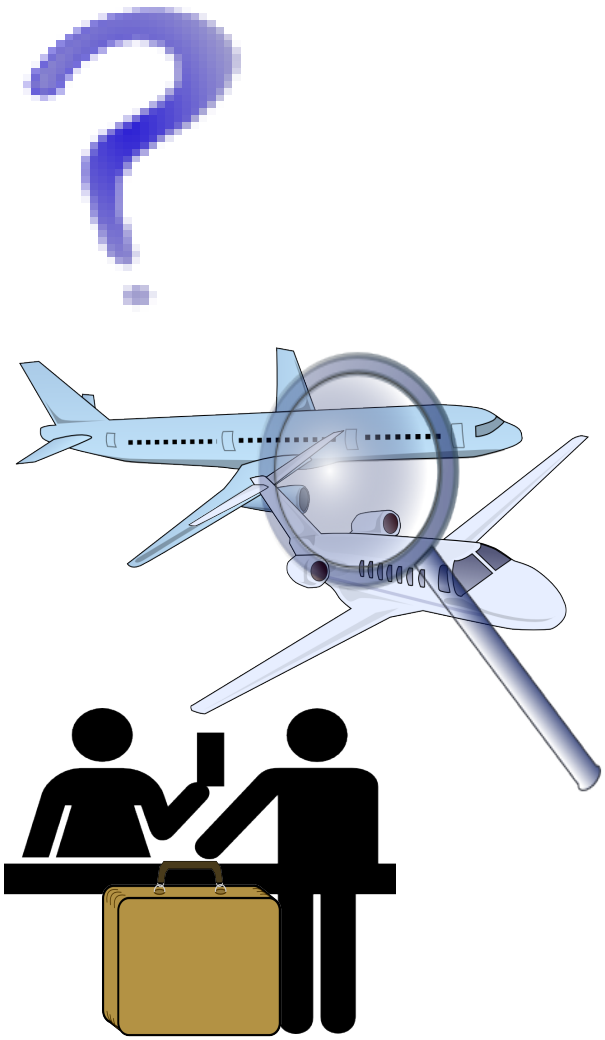
Travel Scenario:

Book a flight from
Frankfurt to Lisboa

Search for flights from FRA to LIS
on November 01.

Are there available seats for that
flight?

Book a seat for flight LH4532 on
November 01 for passenger
John Doe.



ECA Rules and Semantic Web Services

Conventional webservice:

- Interface description
- Explicit service invocation

Event

book-travel-request

passenger	JohnDoe
from	FRA
to	LIS
date	20091101

Condition

book-travel-request

passenger	JohnDoe
from	FRA
to	LIS
date	20091101

```
Select ?connection
where{
        ?connection is a flight
        from    FRA
        to      LIS
        date    20091101}
```

Action

book-travel-request

passenger	JohnDoe
from	FRA
to	LIS
date	20091101

```
Select ?connection
where{
        ?connection is a flight
        from    FRA
        to      LIS
        date    20091101}
```

do-flight-booking

passenger	JohnDoe
flight	http://travel.org/iata/airlines/LufthansaFlight#LH4532-20091101

EventConditionAction (ECA) Rules

```
<eca:Rule>
  <eca:Event>
     <travel:book-travel-request>
       <travel:person>{$Person}</travel:person>
       <travel:from>{$From}</travel:from>
       <travel:to>{$To}</travel:to>
       <travel:date>{$Date}</travel:date>
     </travel:book-travel-request>
  </eca:Event>
  <eca:Query>
       SELECT ?connection
       WHERE  {
          ?connection a travel:flight.
          ?connection  travel:from ?From.
          ?connection  travel:to   ?To.
          ?connection  travel:date $Date.
       }
  </eca:Query>
  <eca:Action>
     <travel:do-flight-booking>
       <travel:passenger>{$Person}</travel:passenger>
       <travel:flight>{$connection}</travel:flight>
     </travel:do-flight-booking>
  </eca:Action>
</eca:Rule>
```

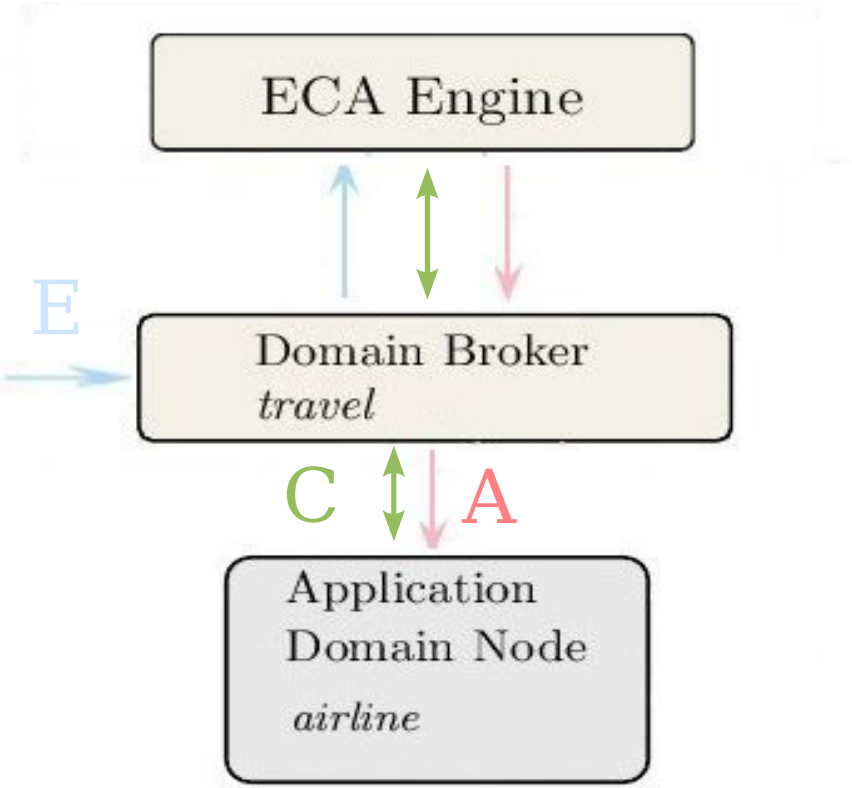
E

C

A

MARS (Modular Active Rules for the Semantic Web)

```
<travel:book-travel-request>
  <travel:passenger>
    http://example.org#JohnDoe
  </travel:passenger>
  <travel:from>FRA</travel:from>
  <travel:to>LIS</travel:to>
  <travel:date>20091101</travel:date>
</travel:book-travel-request>
```



ECA Rules and Semantic Web Services

Conventional webservice:	Event-driven environment:
Interface description	Rule specifications Domain ontology
Explicit service invocation	uniform workflow

Modularity in MARS

Action, condition and event components can be specified in arbitrary languages

e.g. CCS for complex actions, XQuery for conditions, SNOOP for complex events

New language (services) can be added easily

Domain Ontologies

book-travel-request

Passenger	JohnDoe
from	FRA
to	LIS
date	20091101

Domain Ontologies

book-travel-request

Passenger	JohnDoe
from	FRA
to	LIS
date	20091101

```
book-travel-request
                    is-a event ;
                    has parameter { passenger, from, to, date }.
```

Domain Ontologies

```
Select ?connection
where{
      ?connection is-a flight
       from   FRA
       to     LIS
       date   20091101}
```

Domain Ontologies

```
Select ?connection
where{
    ?connection is-a flight
    from  FRA
    to    LIS
    date  20091101}
```

```
book-travel-request
                    is-a event ;
                    has parameter { passenger, from, to, date }.

flight    is-a travel-concept
          has properties  { from, to, date } .
```

Domain Ontologies

do-flight-booking

passenger	JohnDoe
flight	http://travel.org/iata/airlines/LufthansaFlight#LH4532-20091101

Domain Ontologies

do-flight-booking

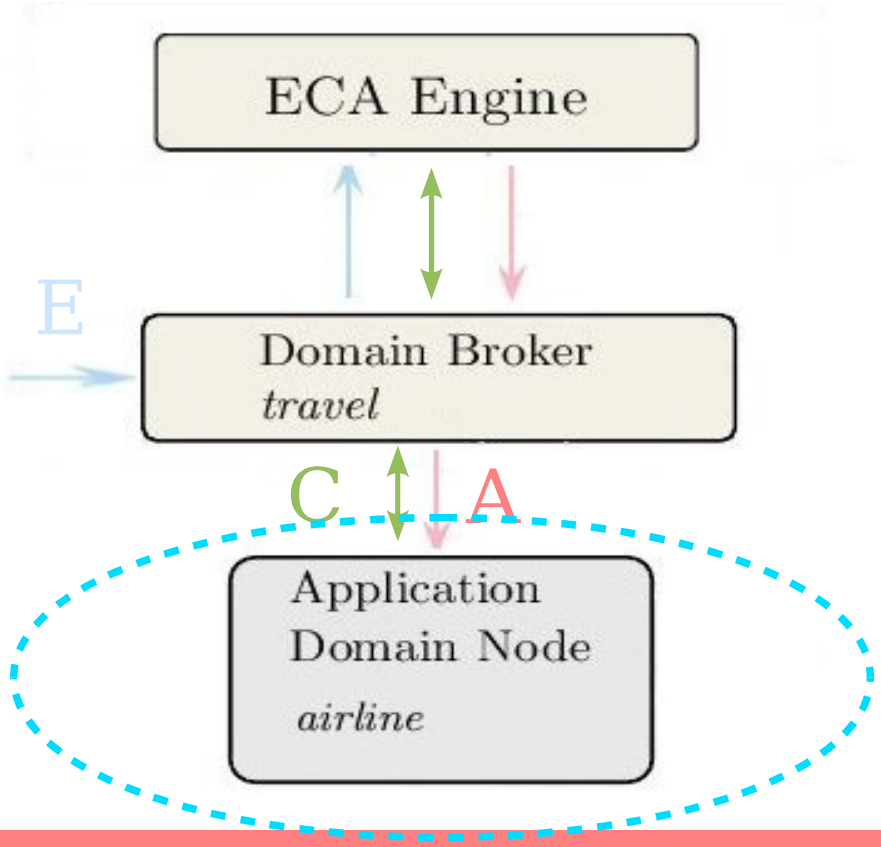
passenger JohnDoe
flight http://travel.org/iata/airlines/LufthansaFlight#LH4532-20091101

```
book-travel-request
                    is-a event ;
                    has parameter { passenger, from, to, date }.

flight   is-a travel-concept
         has properties  { from, to, date } .

do-flight-booking   is-a action ;
                    has parameter { passenger, served-flight }.
```

Domain Ontologies

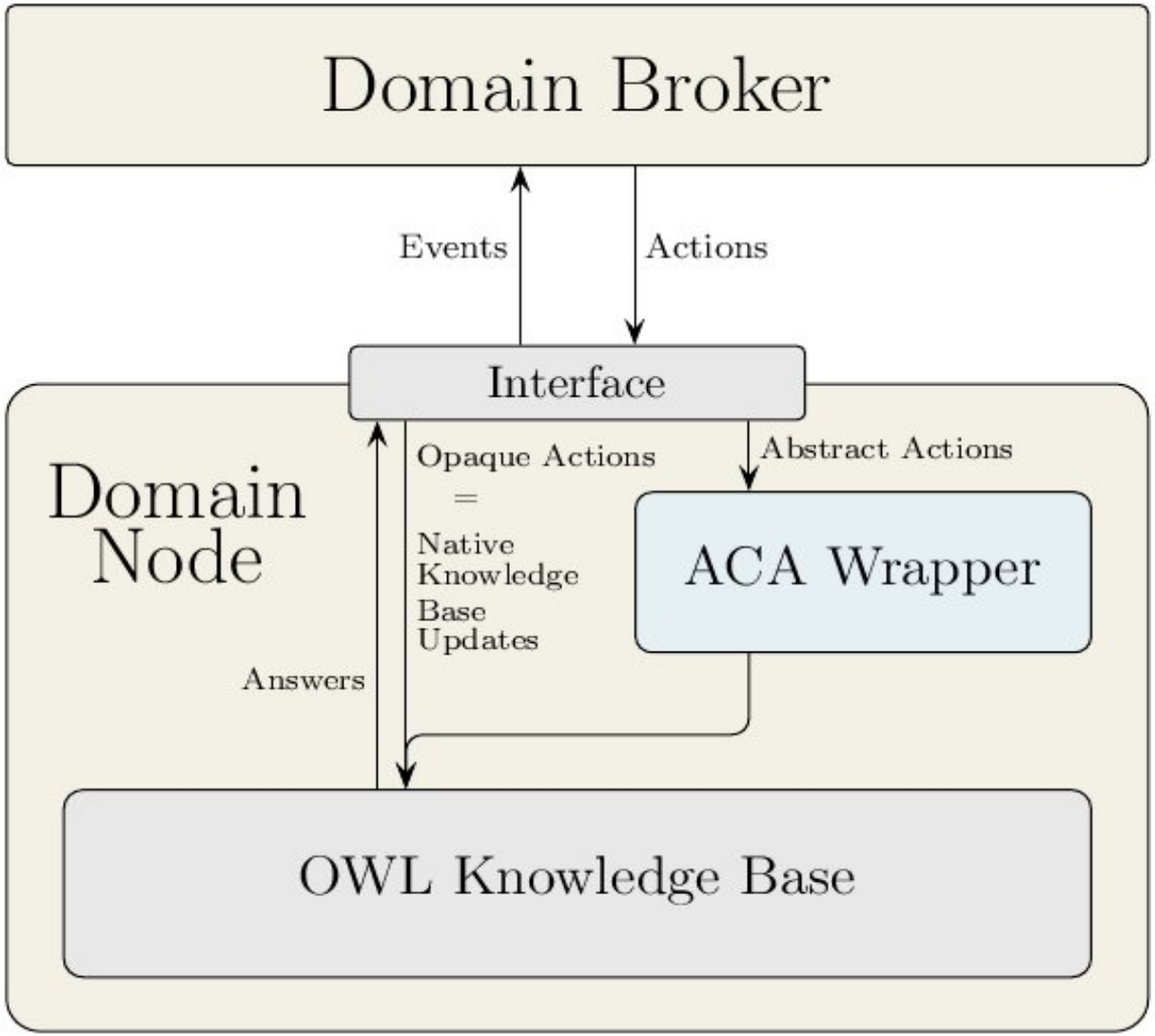
```
book-travel-request
                    is-a event ;
                    has parameter { passenger, from, to, date }.

flight    is-a travel-concept
          has properties  { from, to, date } .

do-flight-booking    is-a action ;
                    has parameter { passenger, served-flight }.
```

MARS and SWAN

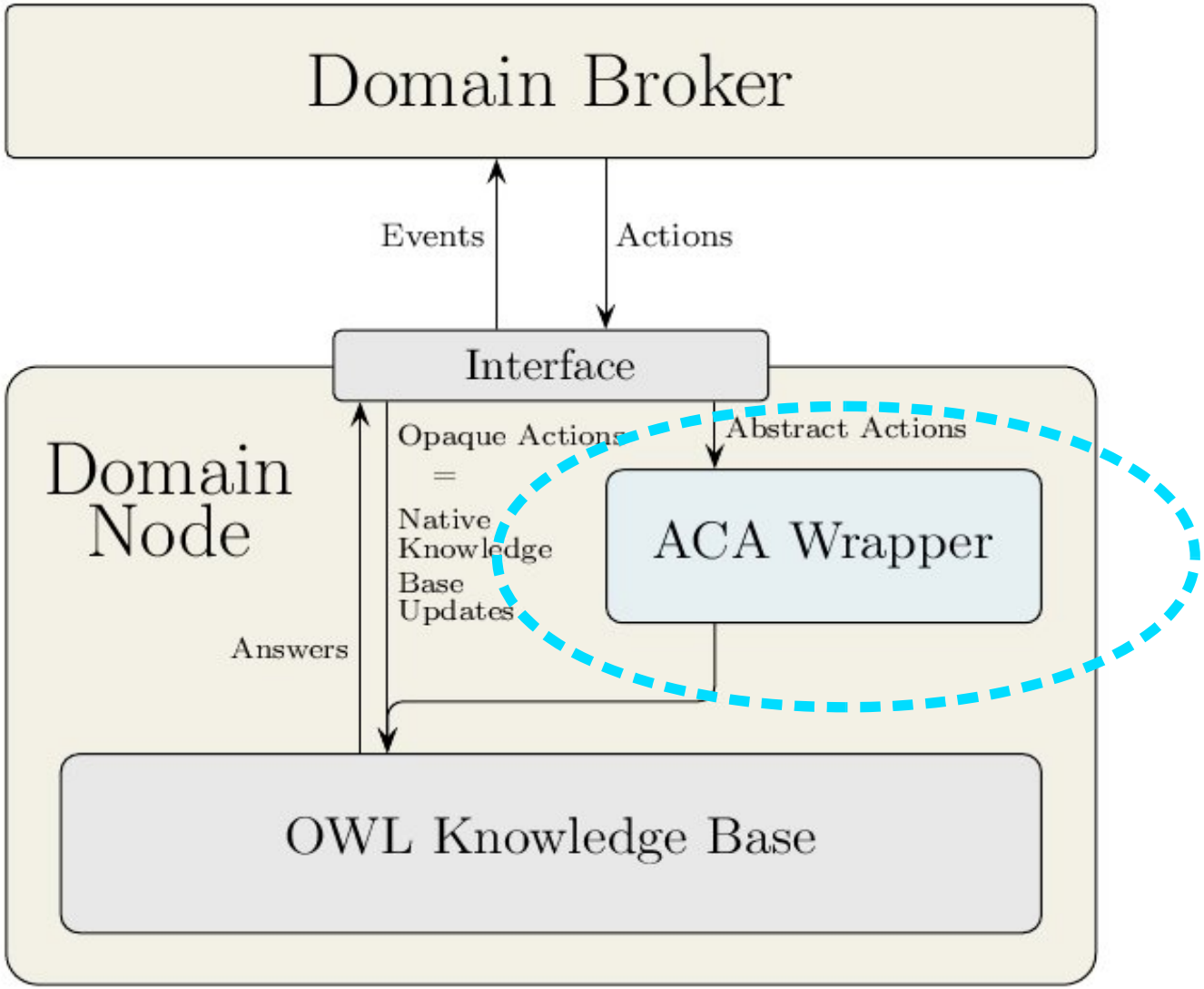



```
<travel:do-flight-booking>
  <travel:passenger>http://example.org#JohnDoe</travel:passenger>
  <travel:flight>http://travel.org/iata/airlines/LufthansaFlight#LH4532-0091101
  </travel:flight>
</travel:do-flight-booking>
```

Node Architecture



Node Architecture



Mapping high-level Actions to KB Updates

High-level Action

```
<travel:do-flight-booking>
  <travel:passenger>http://example.org#JohnDoe</travel:passenger>
  <travel:flight>
   http://travel.org/iata/airlines/LufthansaFlight#LH4532-20091101
  </travel:flight>
</travel:do-flight-booking>
```

Mapping high-level Actions to KB Updates

High-level Action

```
<travel:do-flight-booking>
  <travel:passenger>http://example.org#JohnDoe</travel:passenger>
  <travel:flight>
   http://travel.org/iata/airlines/LufthansaFlight#LH4532-20091101
  </travel:flight>
</travel:do-flight-booking>
```

Local Action (KB command)

```
insert
  ( http://travel.org/iata/airlines/LufthansaFlight#LH4532-20091101,
    http://travel.org#hasBooking ,
    http://example.org#JohnDoe )
```

Mapping high-level Actions to KB Updates

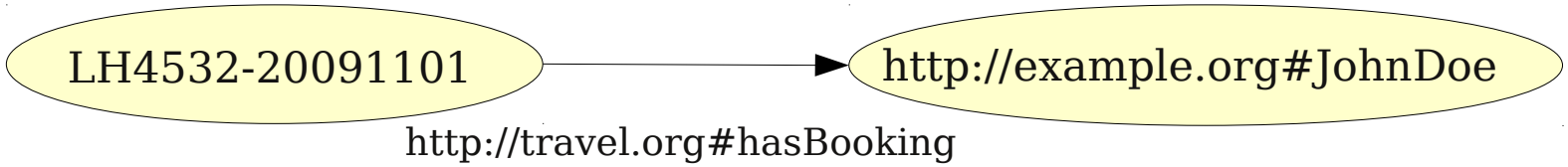
High-level Action

```
<travel:do-flight-booking>
  <travel:passenger>http://example.org#JohnDoe</travel:passenger>
  <travel:flight>
   http://travel.org/iata/airlines/LufthansaFlight#LH4532-20091101
  </travel:flight>
</travel:do-flight-booking>
```

Local Action (KB command)

```
insert
  ( http://travel.org/iata/airlines/LufthansaFlight#LH4532-20091101,
    http://travel.org#hasBooking ,
    http://example.org#JohnDoe )
```

Graph Update



ACA Rules using XQuery

```
for $booking in //travel:do-flight-booking
let $flight := $booking/travel:flight
let $person := $booking/travel:passenger
return
<rdfu:condition
 rdfu:ask="&lt;{$flight}&gt; a travel:served_flight.
   &lt;{$flight}&gt; travel:hasBookings ?booking.
   &lt;{$flight}&gt; travel:hasFlightNo ?con.
   ?con rdf:type travel:Flight .
   ?con travel:withType ?typ.
   ?typ iata:person_capacity ?cap.
   FILTER (?booking &lt; ?cap).
  ">
  <rdfu:insert>
     <rdf:subject rdf:about="{$flight}"/>
     <rdf:predicate rdf:about=
       "http://travel.org/hasBooking"/>
     <rdf:object rdf:about="{$person}"/>
  </rdfu:insert>
</rdfu:condition>
```

Match high-level action
Extract variables

A

ACA Rules using XQuery

```
for $booking in //travel:flight-booking
let $flight := $booking/travel:flight
let $person := $booking/travel:passenger
return
<rdfu:condition
 rdfu:ask="&lt;{$flight}&gt; a travel:served_flight.
   &lt;{$flight}&gt; travel:hasBookings ?booking.
   &lt;{$flight}&gt; travel:hasFlightNo ?con.
   ?con rdf:type travel:Flight .
   ?con travel:withType ?typ.
   ?typ iata:person_capacity ?cap.
   FILTER (?booking &lt; ?cap).
  ">
  <rdfu:insert>
     <rdf:subject rdf:about="{$flight}"/>
     <rdf:predicate rdf:about=
       "http://travel.org/hasBooking"/>
     <rdf:object rdf:about="{$person}"/>
  </rdfu:insert>
</rdfu:condition>
```

Match high-level action
Extract variables

A

Check conditions
Bind local variables

C

ACA Rules using XQuery

```
for $booking in //travel:flight-booking
let $flight := $booking/travel:flight
let $person := $booking/travel:passenger
return
<rdfu:condition
 rdfu:ask="&lt;{$flight}&gt; a travel:served_flight.
   &lt;{$flight}&gt; travel:hasBookings ?booking.
   &lt;{$flight}&gt; travel:hasFlightNo ?con.
   ?con rdf:type travel:Flight .
   ?con travel:withType ?typ.
   ?typ iata:person_capacity ?cap.
   FILTER (?booking &lt; ?cap).
  ">
  <rdfu:insert>
    <rdf:subject rdf:about="{$flight}"/>
    <rdf:predicate rdf:about=
      "http://travel.org/hasBooking"/>
    <rdf:object rdf:about="{$person}"/>
  </rdfu:insert>
</rdfu:condition>
```

Match high-level action
Extract variables

A

Check conditions
Bind local variables

C

Create update statements

A

Components in SWAN: RDFU

RDFU: RDF Update Language

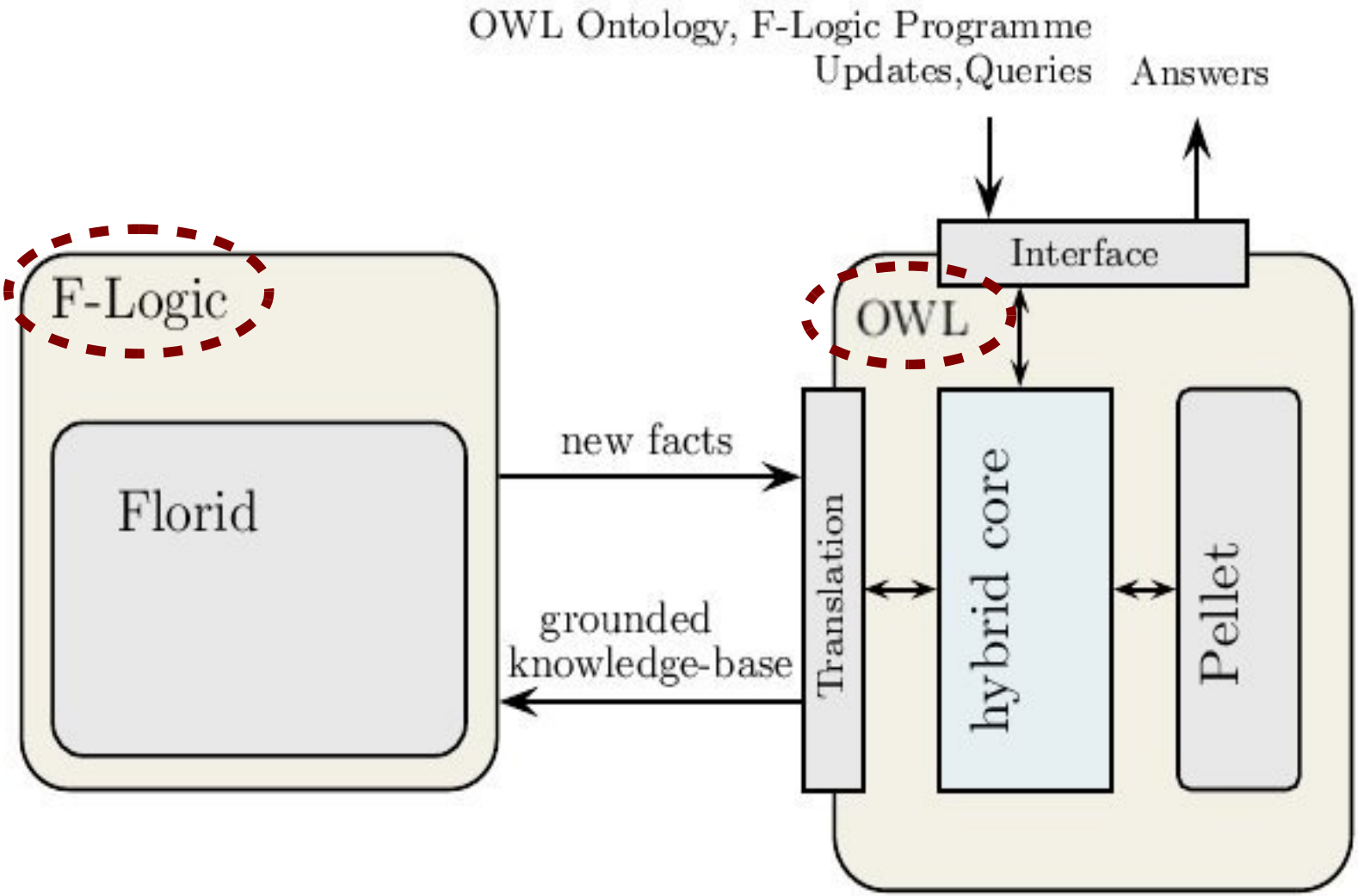
knowledge base updates

support for intensional updates in presence of reasoning

Extending DL Reasoning

Hybrid reasoning combines Description Logics with F-Logic

Hybrid Reasoning in SWAN



Active Knowledge Base

Knowledge Base Triggers:

- update completion
- maintenance of KB integrity
- invocation of hybrid reasoning
- Raising of events

Rule-Based Event-Generation

Knowledge base trigger (abstract syntax)

```
ON INSERTION OF  hasBooking

WHEN
        SELECT ?flightNo ?date FROM KB

DO
BEGIN
  raise event(
    <travel:flightBooked>
        <travel:flightNo>?flightNo</travel:flightNo>
        <travel:passenger>$new.object</travel:passenger>
        <travel:date>?date</travel:date>
    </travel:flightBooked>
  );
END;
```

E

C

A

Rule-Based Event-Generation

Knowledge base trigger (abstract syntax)

```
ON INSERTION OF  hasBooking

WHEN
        SELECT ?flightNo ?date FROM KB

DO
BEGIN
  raise event(
    <travel:flightBooked>
        <travel:flightNo>?flightNo</travel:flightNo>
        <travel:passenger>$new.object</travel:passenger>
        <travel:date>?date</travel:date>
    </travel:flightBooked>
  );
END;
```

E

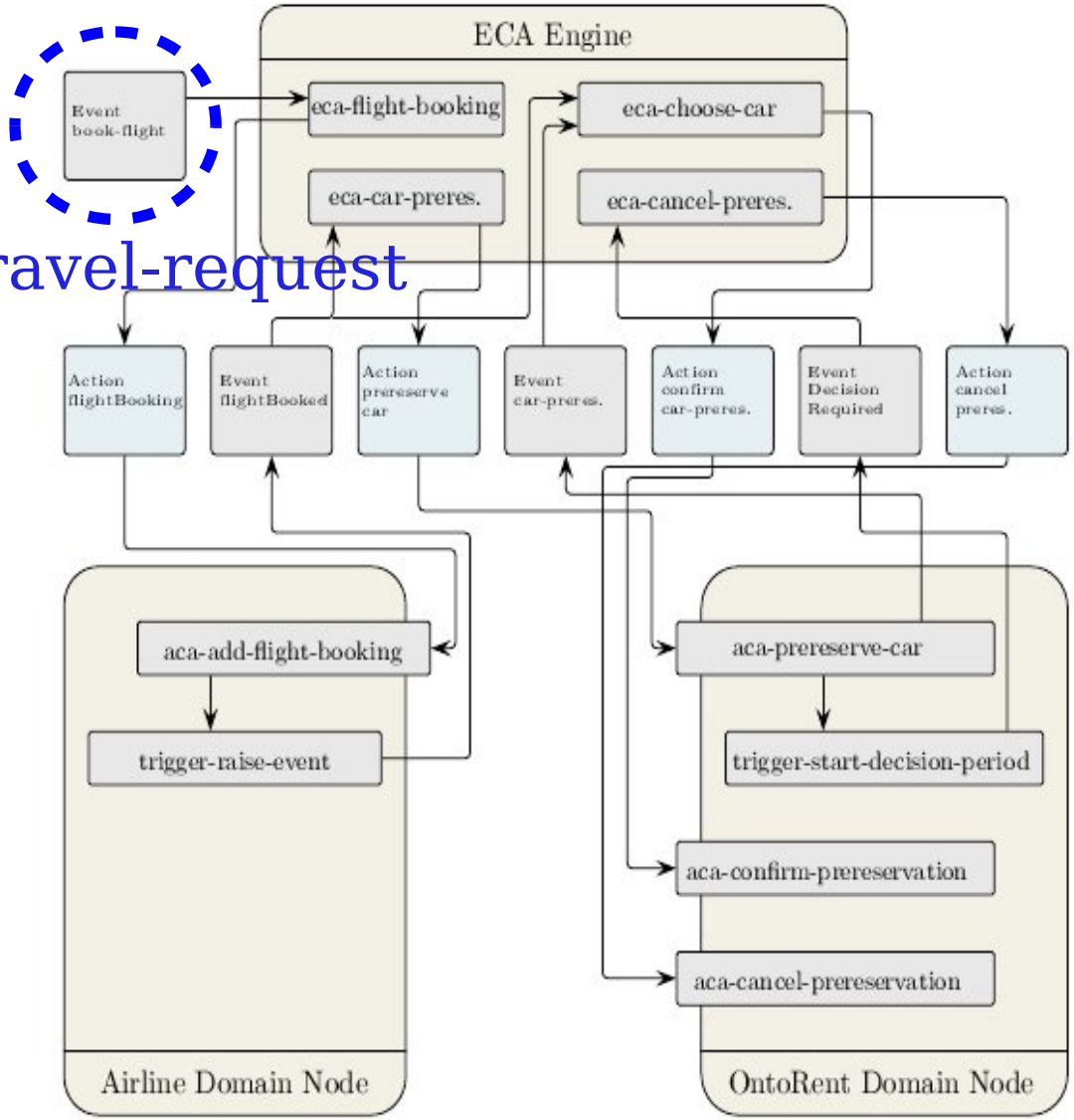
C

A

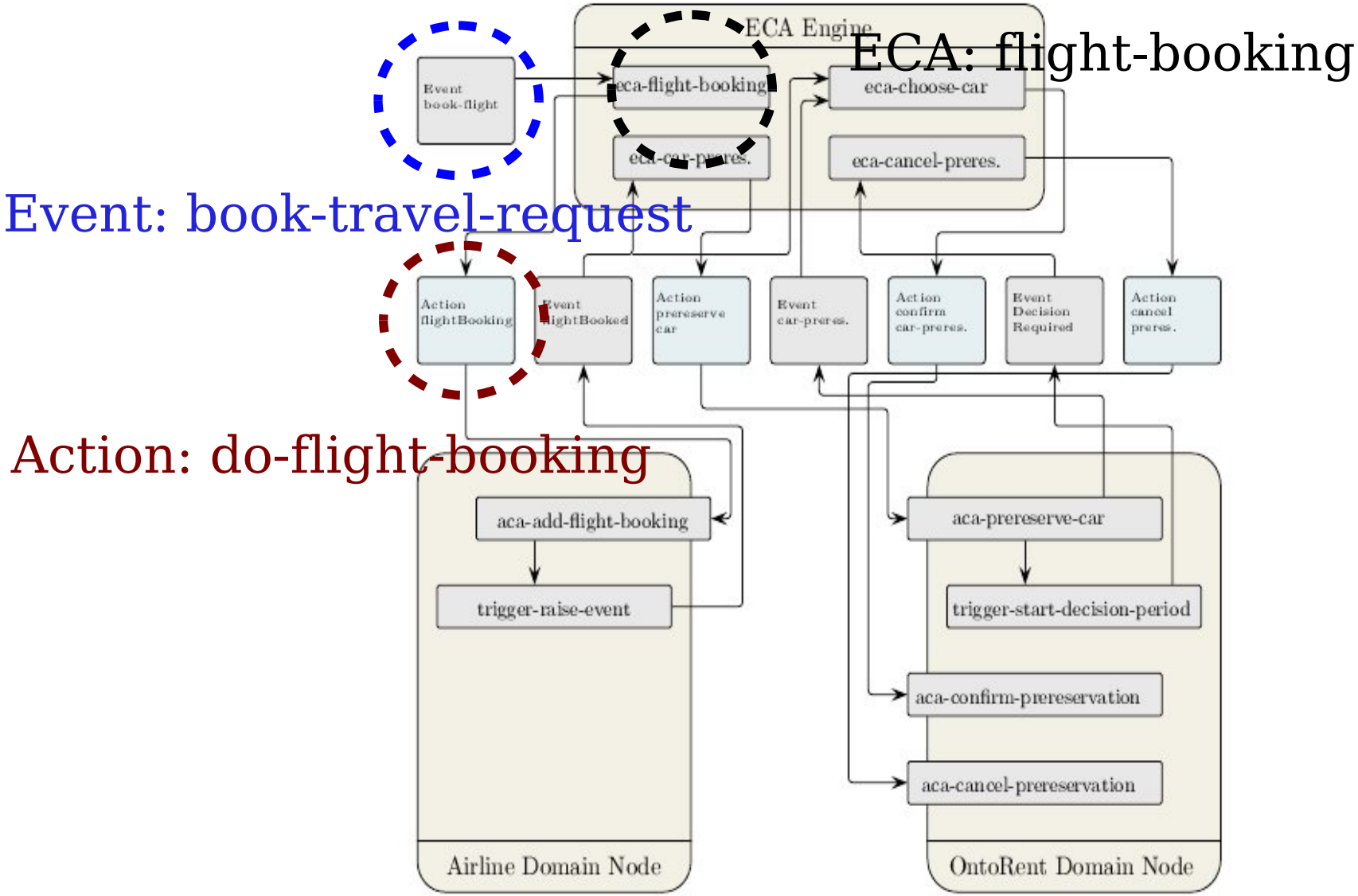
E

Travel Scenario: Simple Workflow

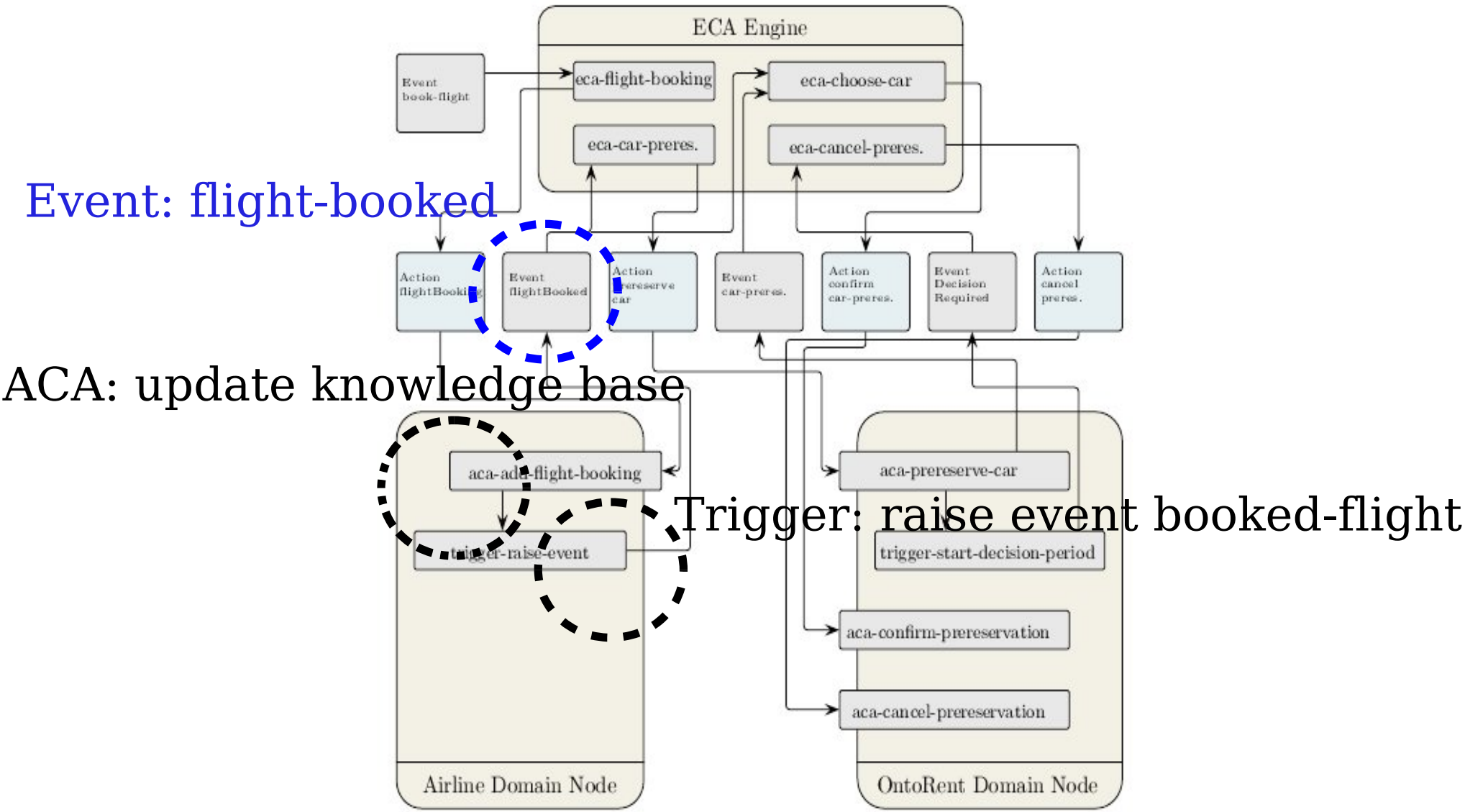
Event: book-travel-request



Travel Scenario: Simple Workflow



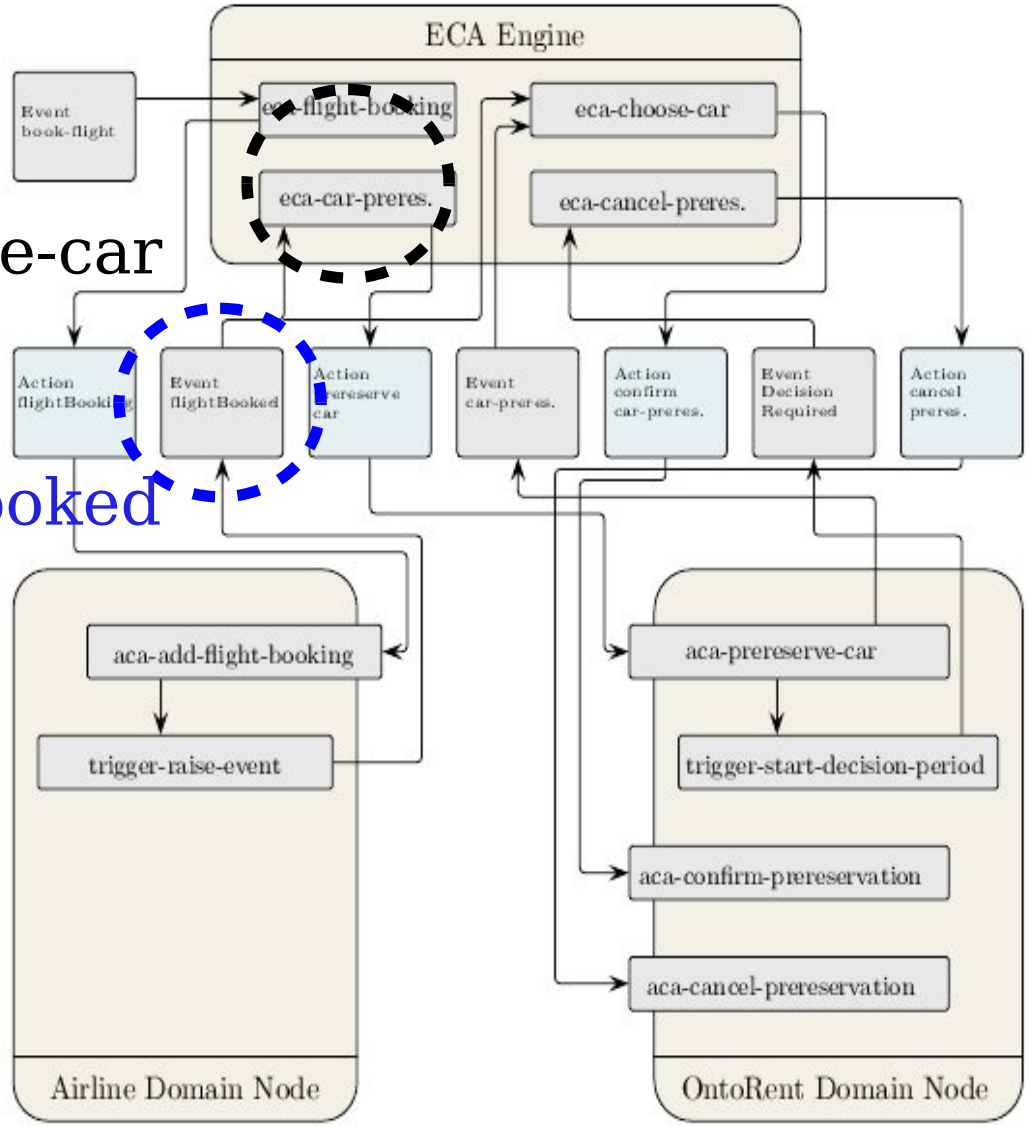
Travel Scenario: Simple Workflow



Travel Scenario: Simple Workflow

ECA: prereserve-car

Event: flight-booked



Summary

Event-driven service environment:
interface is invisible to the user

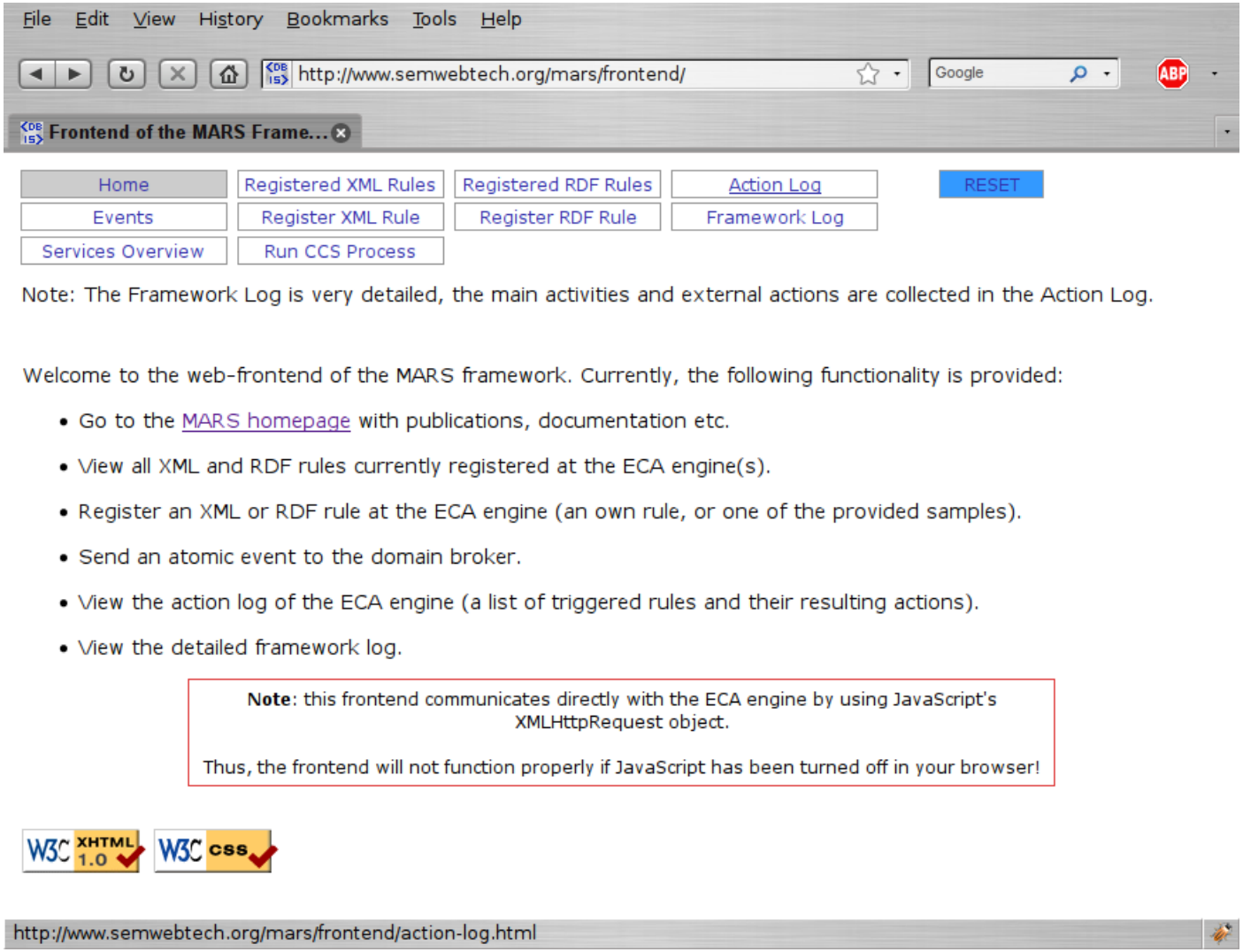
Extensible: new language services and application nodes can be deployed easily.

All behaviour is specified by reactive rules
Global ECA rules, ACA rules, Local ECA rules(triggers)

No implementation necessary. Only rules have to be deployed to a new copy of a domain node instance.

Implementation

http://www.semwebtech.org/mars/frontend/



SWAN Demonstrator

http://www.semwebtech.org/swan/frontend/

File Edit View History Bookmarks Tools Help

http://www.semwebtech.org/swan/frontend/

Domain Frontend

Travel Scenario Frontend

Application Domain Nodes

The following domain nodes are part of the travel domain:

- SWAN01 - Airline company
- SWAN02 - OntoRent - Car Rental
- SWAN03 - Testbed

Node Administration

see HOWTO for some explanations.

reset — Reset selected Node(s)

upload — Upload Data to selected node(s) from URL:

upload-url

N3 RDF/XML

listtriggers — List registered Triggers

Registered Action Patterns

Raise Direct Event / Send Action

Predefined Patterns

---------- Events ----------
event-flight-booking.xml
event-car-prereservation.xml
event-ant-move.xml
---------- Actions ----------
action-book-flight.xml
action-cancel-flight-booking.xml
action-cancel-flight.xml
action-book-car.xml
action-reserve-car.xml
action-cancel-car-prereservation.xml
action-ant-changeColour.xml
action-ant-addSquare.xml
---------- ACA rules ----------
aca-add-flight-booking.xml
aca-cancel-flight-booking.xml
aca-cancel-flight.xml
aca-prereserve-car.xml
aca-cancel-car-prereservation.xml
aca-confirm-car-prereservation.xml
aca-ant-change-colour.xml
aca-ant-new-square.xml

Done

Q & A

Thank you!