Lprm

Laboratorio de Pesquisa em Redes e Multimídia

Departamento de Informática

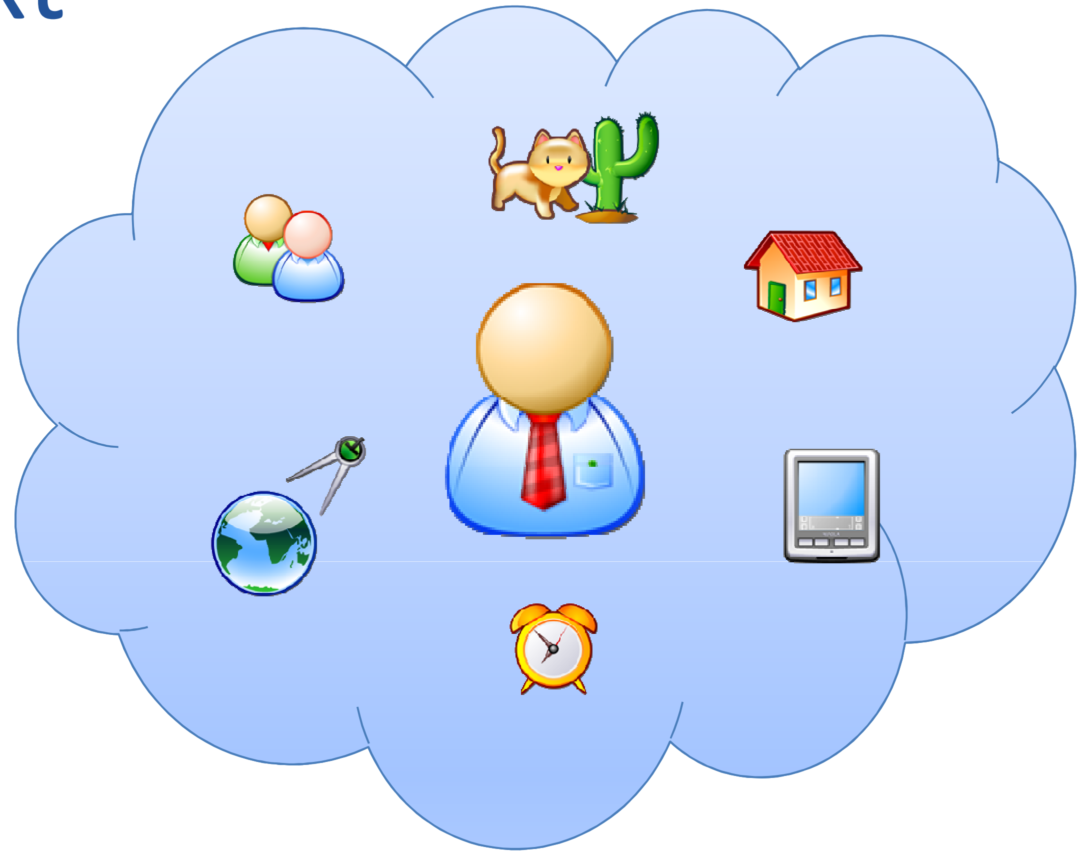
ECA-DL e Transformações

Patrícia Dockhorn Costa

pdcosta@inf.ufes.br

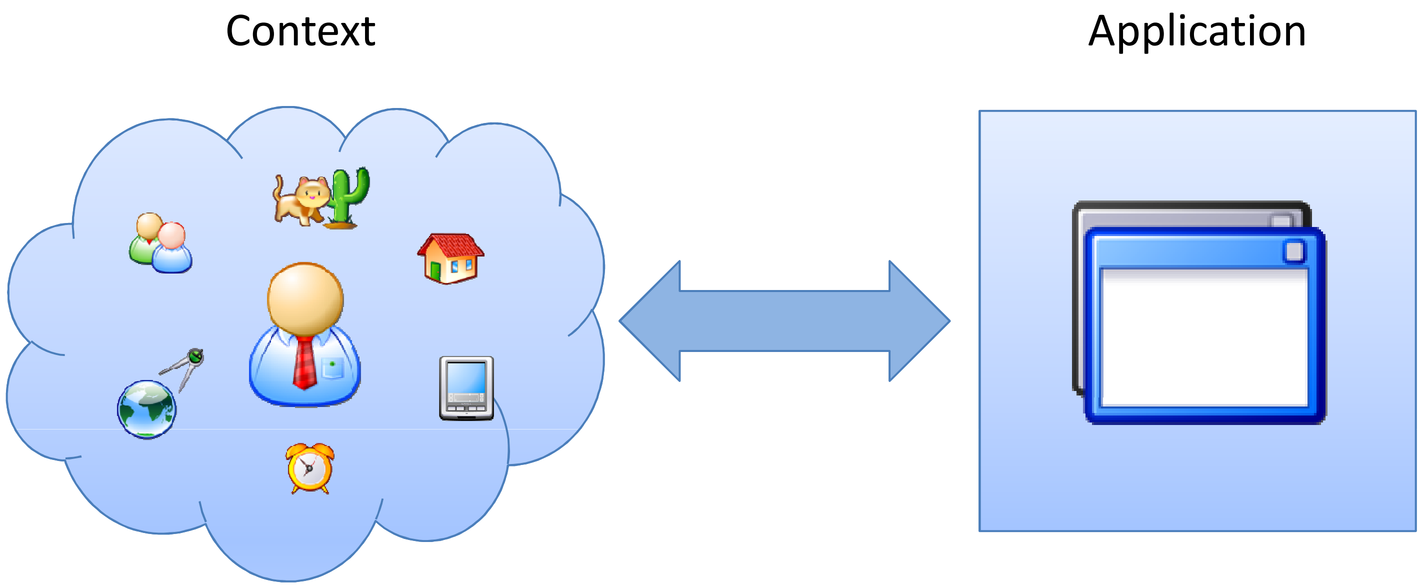
Context-Awareness

Context



the set of possibly interrelated conditions in which an entity exists.

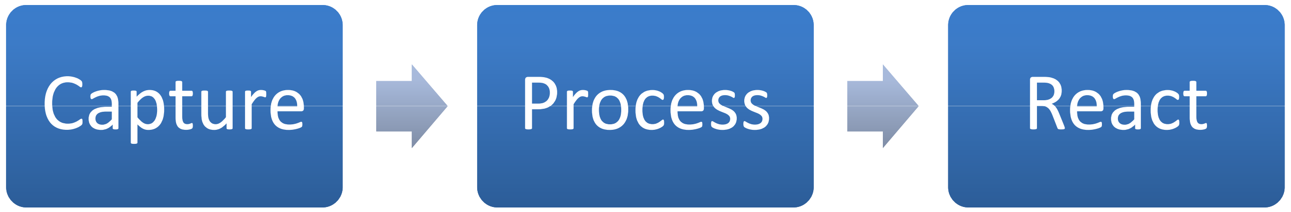
Context-Aware Application



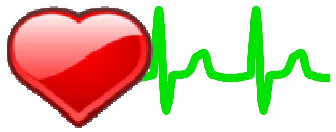
Context-aware application is a **distributed** application whose behaviour is **affected by** its users' **context.**

Developing Context-Aware Applications

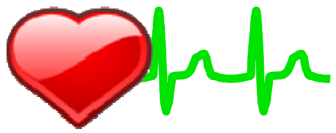
Basic Requirements

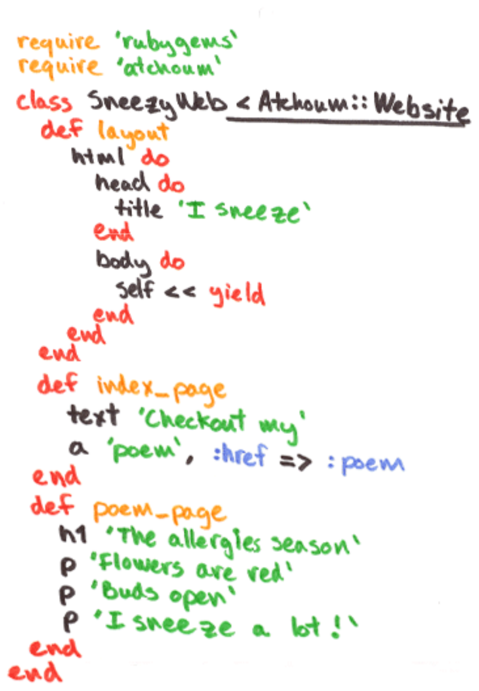


Capture Context



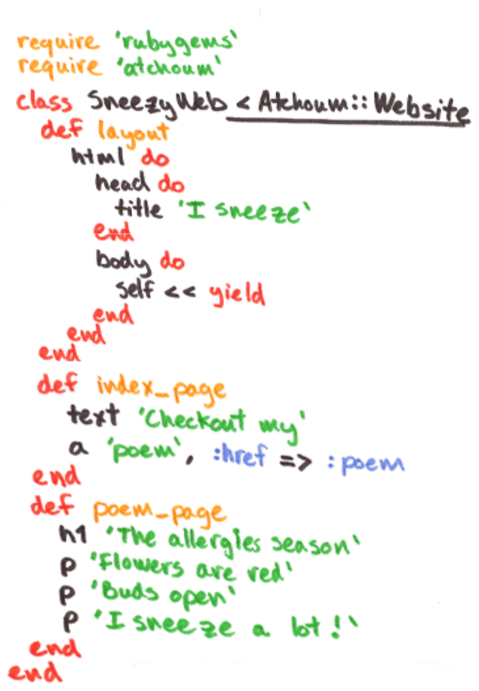
Model Behaviour





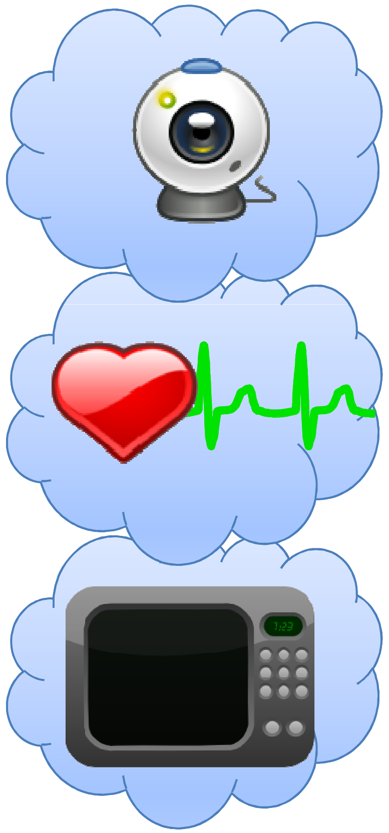
```
require 'rubygems'
require 'atchoum'
class SneezyWeb < Atchoum::Website
  def layout
    html do
      head do
        title 'I sneeze'
      end
      body do
        self << yield
      end
    end
  end
  def index_page
    text 'Checkout my'
    a 'poem', :href => :poem
  end
  def poem_page
    h1 'The allergies season'
    p 'Flowers are red'
    p 'Buds open'
    p 'I sneeze a lot!'
  end
end
```

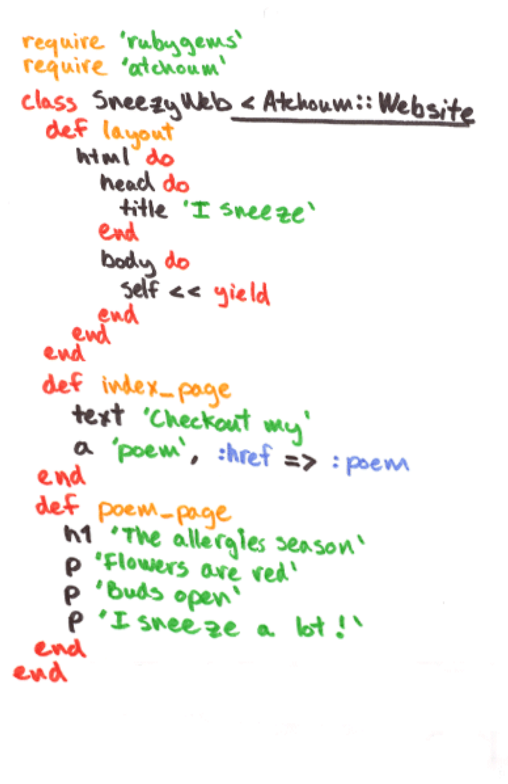
Multitude of Sensors



```
require 'rubygems'
require 'atchoum'
class SneezyWeb < Atchoum::Website
  def layout
    html do
      head do
        title 'I sneeze'
      end
      body do
        self << yield
      end
    end
  end
  def index_page
    text 'Checkout my'
    a 'poem', :href => :poem
  end
  def poem_page
    h1 'The allergies season'
    p 'Flowers are red'
    p 'Buds open'
    p 'I sneeze a lot!'
  end
end
```

Distributed Application

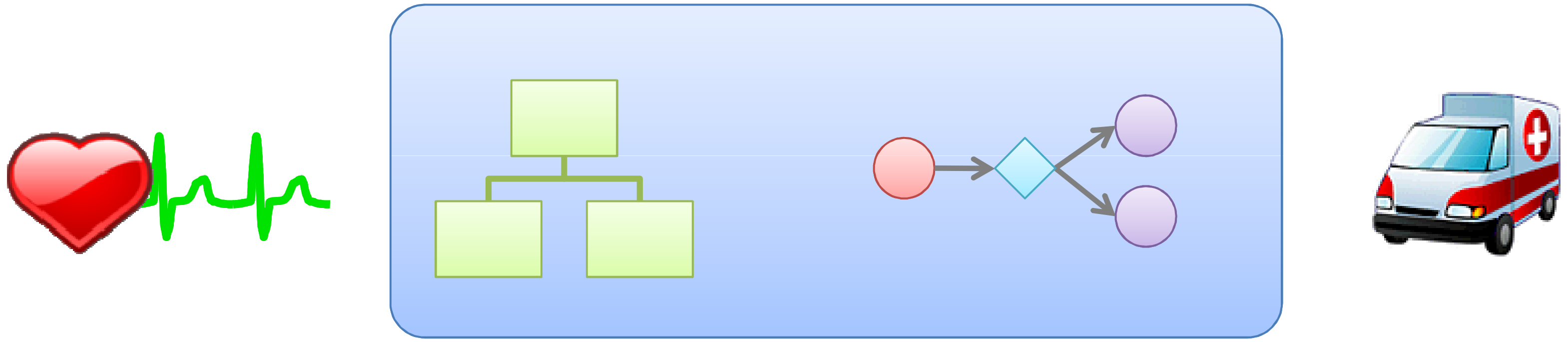




```
require 'rubygems'
require 'atchoum'
class SneezyWeb < Atchoum::Website
  def layout
    html do
      head do
        title 'I sneeze'
      end
      body do
        self << yield
      end
    end
  end
  def index_page
    text 'Checkout my'
    a 'poem', :href => :poem
  end
  def poem_page
    h1 'The allergies season'
    p 'Flowers are red'
    p 'Buds open'
    p 'I sneeze a lot!'
  end
end
```

Context-Aware Application

Supporting Context-Aware Application Development

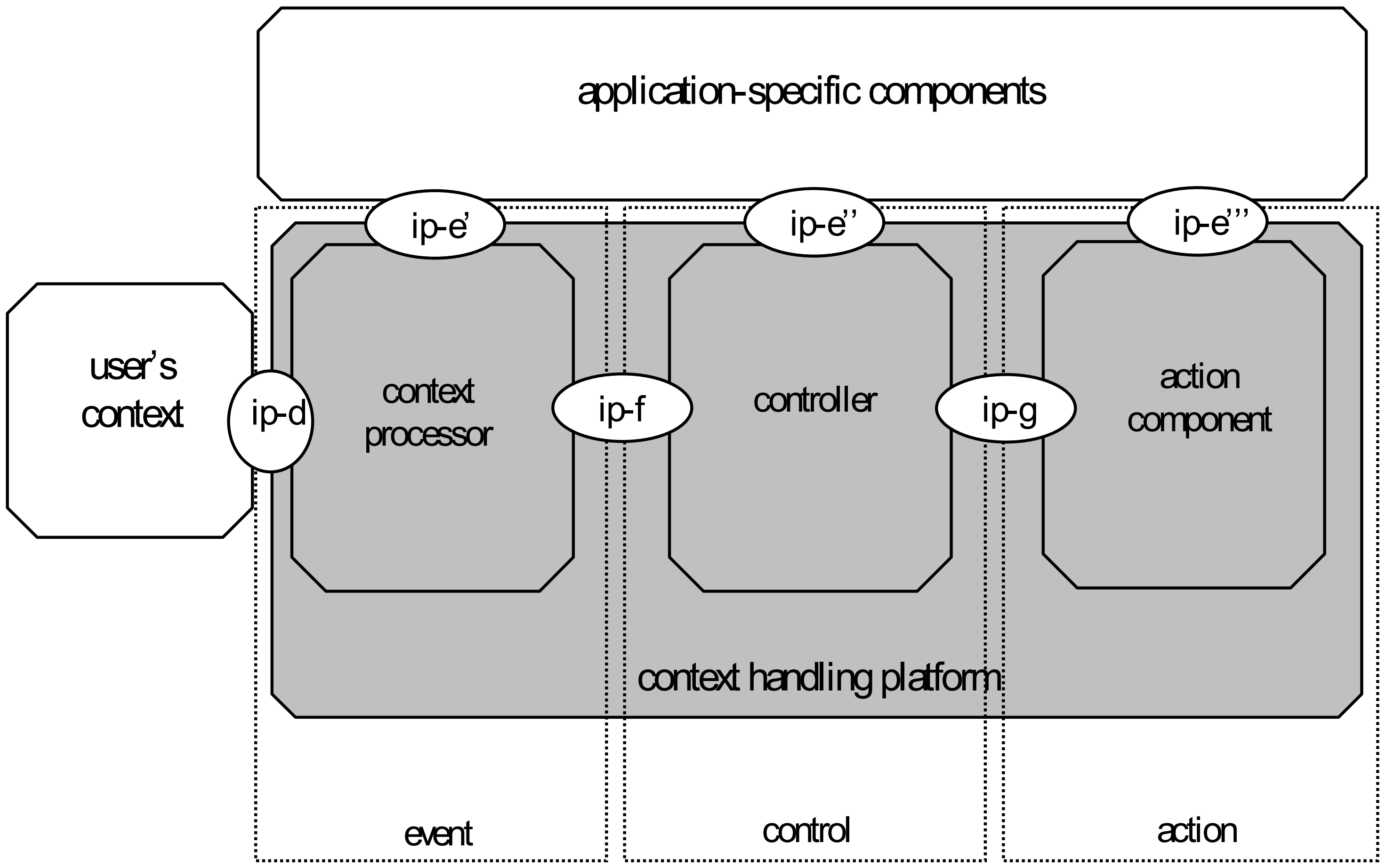


Supporting Context-Aware Application Development (cont.)

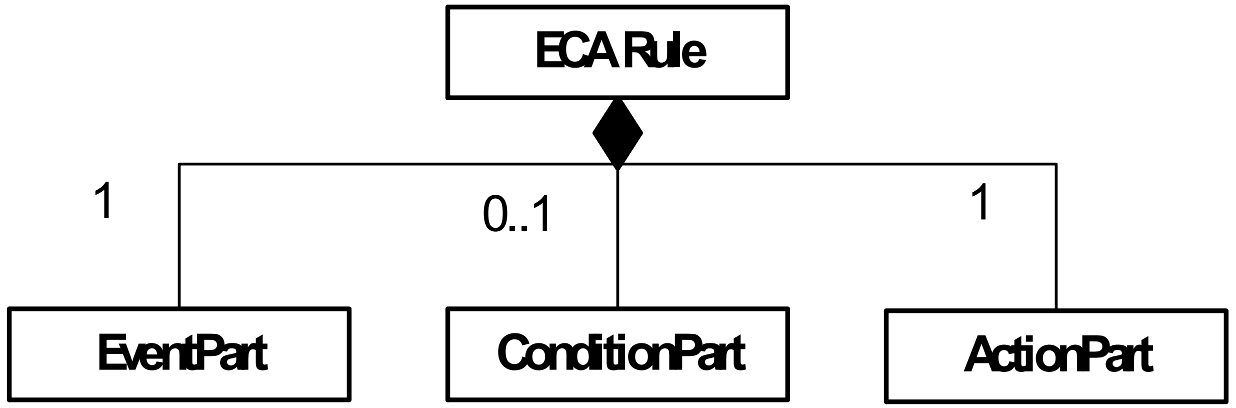
- Quite some work on that:
  - context-aware patterns;
  - services platform (context sources, managers, controllers, action resolvers, etc.);
  - context and situation models;
  - situation reasoning and (distributed) detection;
  - ECA language for modelling reactive behaviours, etc.

ECA-DL

Padrão Event-Control-Action Pattern



Elementos Básicos de ECA-DL



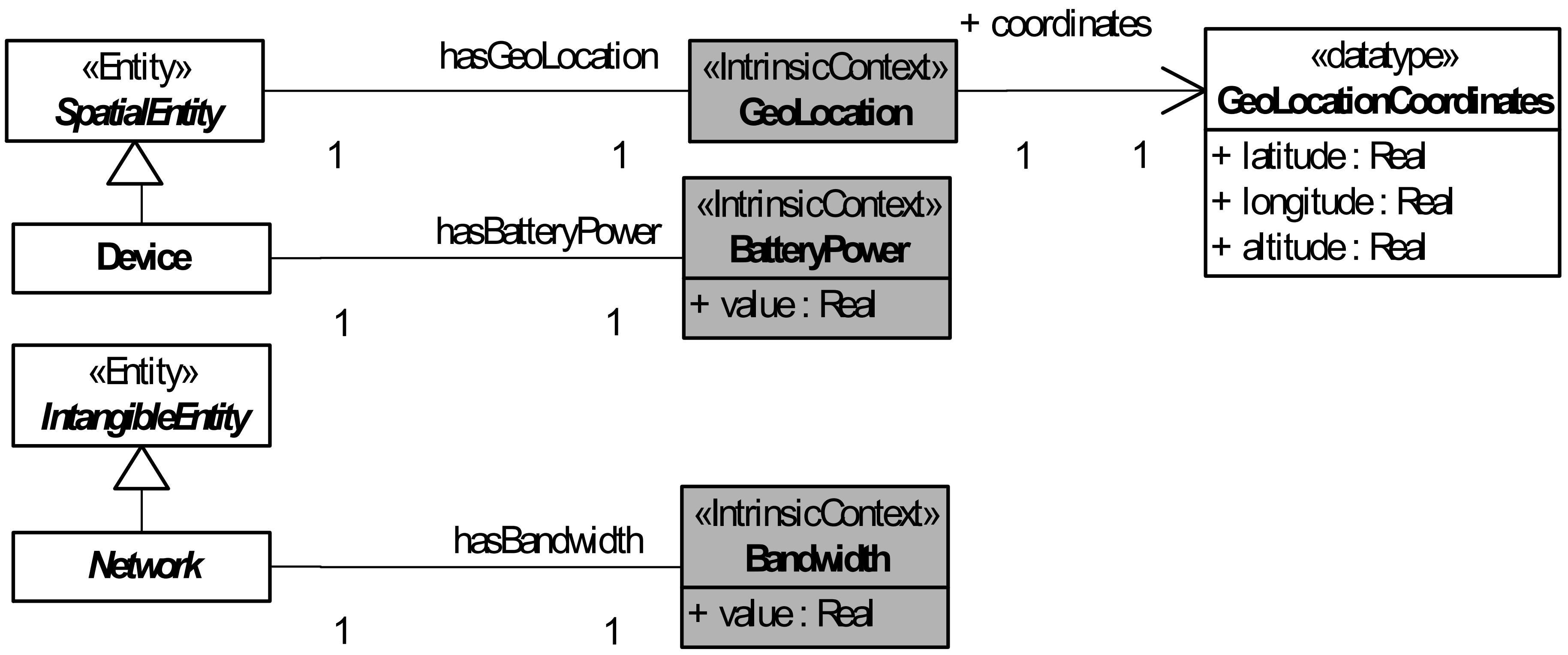
Navigation

- Aims at reaching values or objects of concern in the context and situation models
- Navigation in ECA-DL is similar to navigation in OCL
  - “dots” to navigate from objects to attributes
- The target element is always a primitive datatype (numeric, boolean or string)
- We also include the type of object being navigated

  - EntityType.entityId

Navigation (example)

- Device.id1.hasBatteryPower.value
- Device.id1.hasBandwidth.value
- Device.id1.hasGeoLocation.coordinates.latitude



Navigation - collection

- Person.*
- Supported by means of select clause

Events in ECA-DL

- Situation events (transitions EnterTrue (S) e EnterFalse(S))
  - *EnterTrue (SituationFever (Entity.John))*
- Primitive events: not detected by means of situations
  - *IncomeCall (entityFrom, entityTo)*
- Temporal events
  - Generated from time-to-time
  - *OnEvery(t)*
- Complex events
  - Composition of primitive or situation events

Complex Events

Operator	Composite event
e1&e2	Occurs when both e1 and e2 occur irrespective of their order
{e1;e2} ! e3	Occurs when e1 occurs followed by e2 and e3 does not occur between them
e1|e2	Occurs when e1 or e2 occurs
e1;e2	Occurs when e1 occurs before e2

EnterTrue (SituationContained (Room.3040, Person.John)) ; EnterFalse (SituationContained (Room.3040, Person.John))

Timestamps

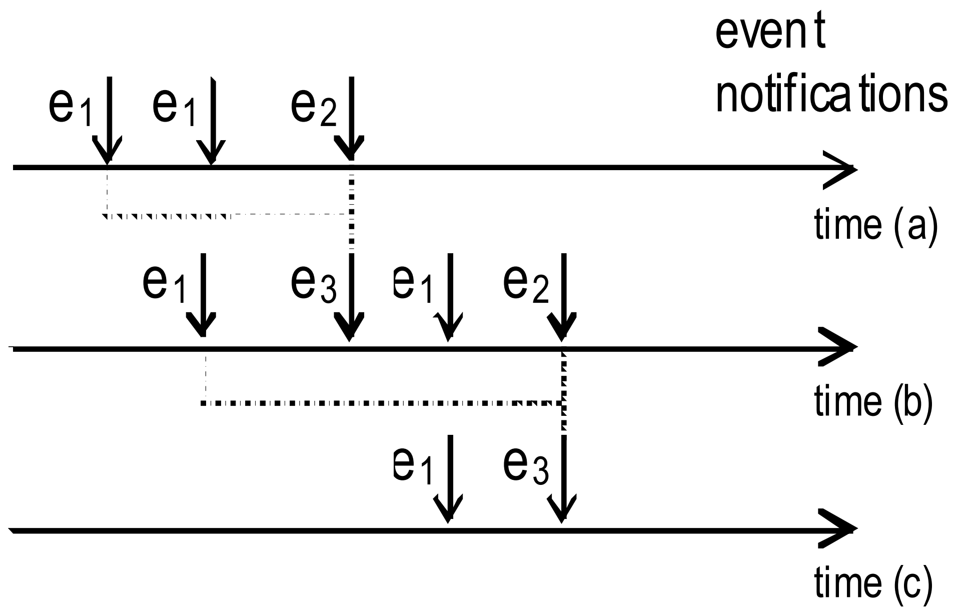
- An event occurs at a specific time: *occurrence time*
- *Occurrence interval* : the time interval during which a composite event is being detected
  - Initiator: initiates the composite detection
  - Terminator: terminates the detection
- The occurrence time of a composite/complex event is the terminator's occurrence time

Event Consumption

- Events are *consumed* by ECA-DL rules
- "John enters the room and at most 15 minutes later he turns his computer on"
- John enters the room at 15:05 hrs, 15:10 hrs, and at 15:13 hrs
- John turns his computer on at 15:15 hrs (terminator)
  - Which notification is the initiator?
  - In ECA-DL, the oldest (15:05)
  - And this notification is NOT considered again: Notification is *consumed* by the rule!
- John turns his computer on again at 15:18 hrs
  - Which notification is the initiator?

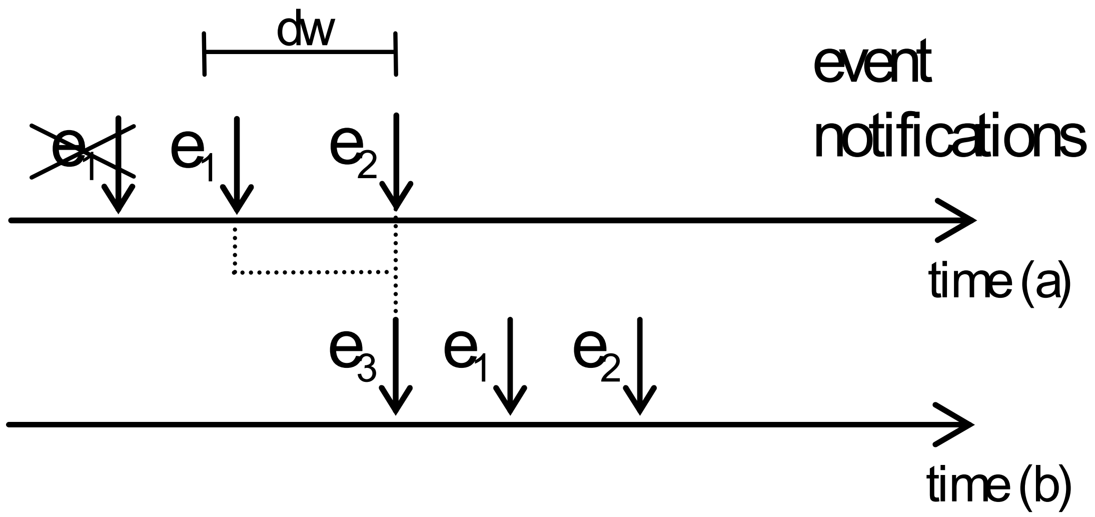
Event Consumption

- Event e3: (e1&e2)
- Notifications: e1, e1, e2, e1, e2



Detection Window

- Event: (e1&e2)
- Notifications: e1, e1, e2, e1, e2



Upon-When-Action clause

Upon <uponExpression>
When <conditionExpression>
Do <actionExpression>

Select clause

Select (<collection-of-entities>; <var>; <filtering-expression-involving-var>)

Select (Person.*, p, p.age>40)
Select (Person.*, p, SituationWithinRange (p))
Select (Select (Person.*, p1, p1.age>40), p2, SituationWithinRange (p2))

Scope clause

```
Scope (<collection-of-entities>; var)
{
   Upon < eventExpression >
   When <conditionExpression>
   Do <actionExpression>
}


Scope (Select (Person.*, person, SituationContained
(person, person.house)), p))
{
    Upon EnterTrue (SituationContained ( p.house))
    Do Notify (p, “there is someone entering the house”)
 }
```

Lifetime

Upon EnterTrue (SituationContained
(Person.John, Person.Mary.house))
Do Notify (Person.Mary, “Mary, John has entered
the house”)

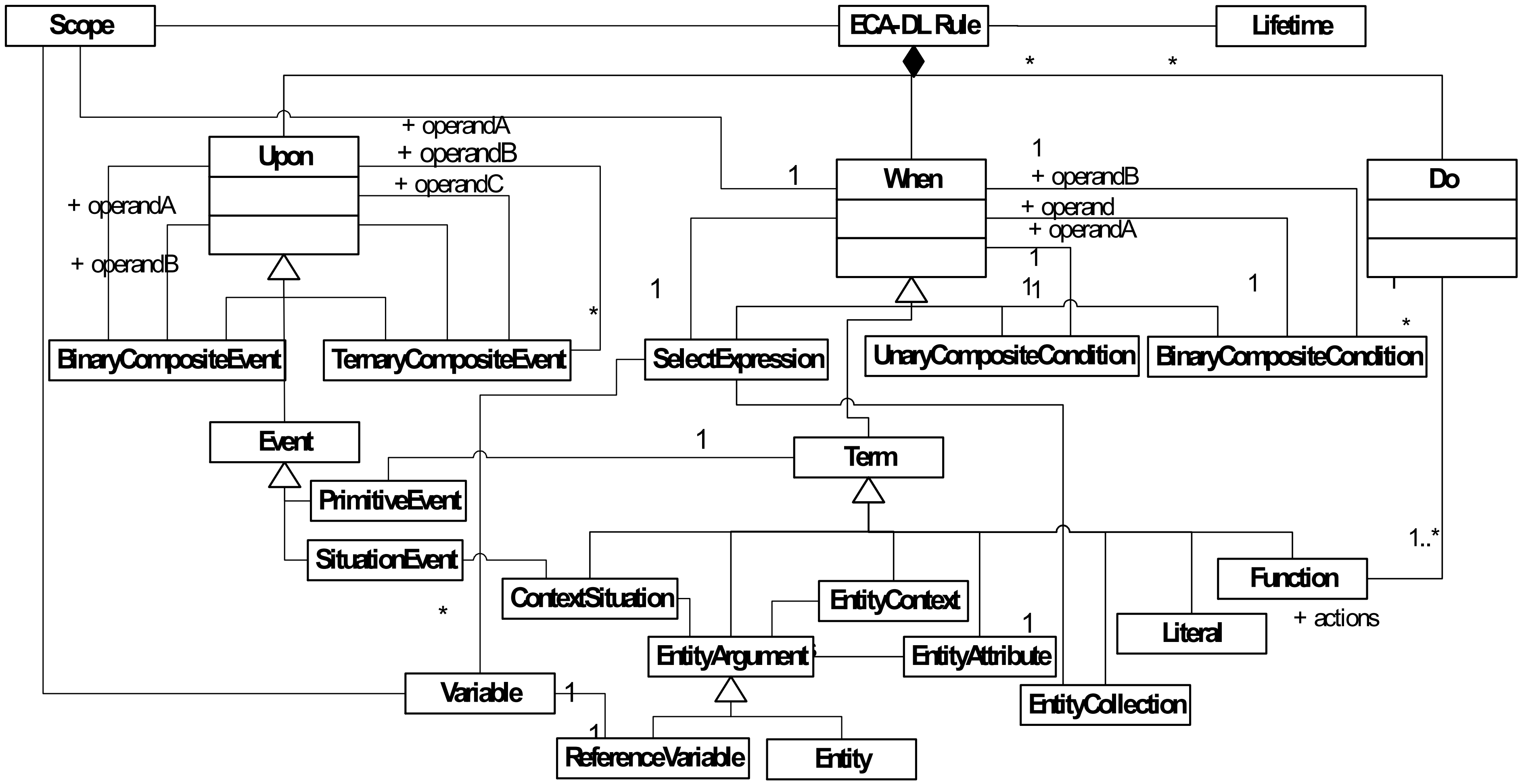
-once

-always

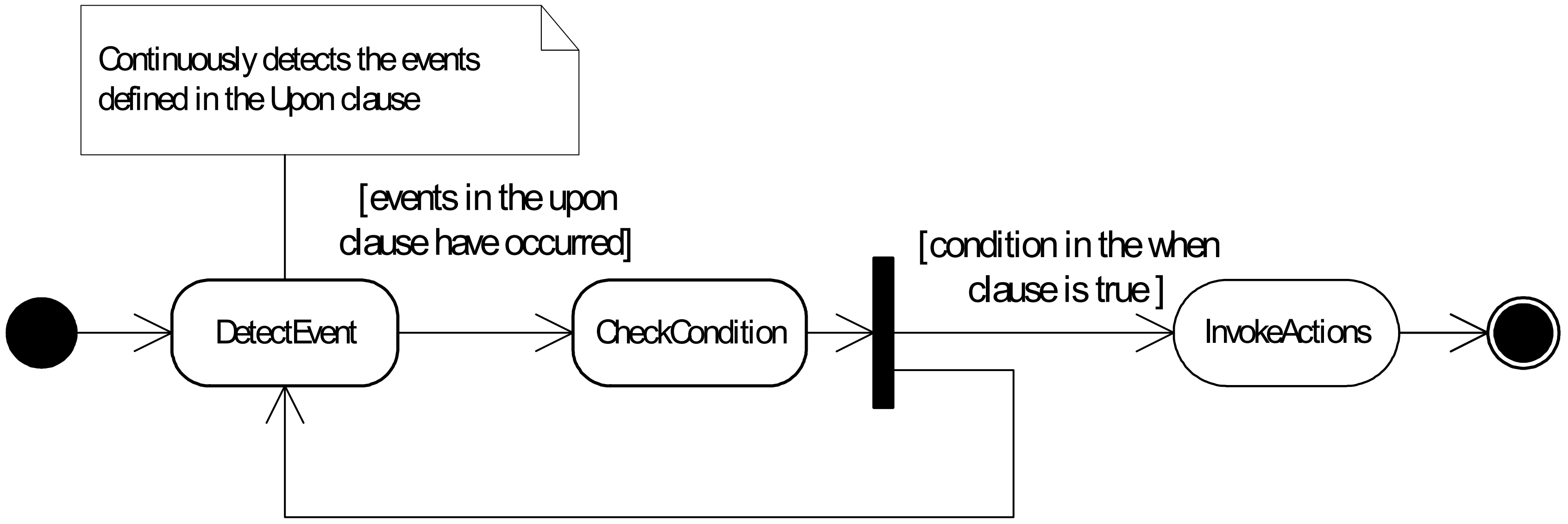
-from<start>to<end>

-etc.

Metamodelo


Scope
ECA-DL Rule
Lifetime
*
*
+ operandA
+ operandB
+ operandC
Upon
+ operandA
+ operandB
1
When
1
+ operandB
+ operand
+ operandA
1
11
1
Do
*
1
BinaryCompositeEvent
TernaryCompositeEvent
*
SelectExpression
UnaryCompositeCondition
BinaryCompositeCondition
Event
1
Term
PrimitiveEvent
SituationEvent
1..*
Function
+ actions
ContextSituation
EntityContext
*
1
Literal
EntityArgument
EntityAttribute
Variable
1
1
EntityCollection
ReferenceVariable
Entity

Rule Execution



Continuously detects the events defined in the Upon clause
[events in the upon clause have occurred]
[condition in the when clause is true]
DetectEvent
CheckCondition
InvokeActions

Examples

```
Upon EpilepticAlarm (Patient.John)
When SituationDriving (Patient.John)
Do SendSms (Patient.John, “John, you may have an epileptic
seizure, please stop the car”)


Scope (select (Patient.*; patient; patient.type = “epileptic” and
patient.hasCivilLocation.city = “Enschede”) ; p)
{
        Upon EpilepticAlarm (p)
        When SituationDriving (p)
        Do SendSms (p, “You may have an epileptic seizure,
please stop the car”)
}
```

Examples

```
Scope (select (Patient.*; patient; patient.type = “diabetic”) ; p)
{
        Upon HighSugarAlarm (p)
        Do SendSms (p, “You have high sugar levels”)
}


Scope (Select (Policeman.*; policeman;
                   policemen.hasActivity.value = “working”; p1)
{
    Upon OnEvery (5)
    Do NotifyApp (application-address,
        List (p1.id, select (policeman.*, p2;
                      SituationWithinRange (p1, p2) and
                      p2.hasActivity.value = ‘working’)))
}
```

ECA-DL to Jess

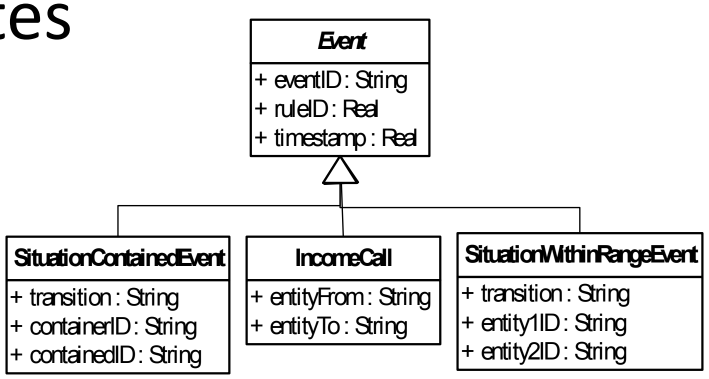
Gerenal Mapping

ECA-DL rule	Jess rule 1	Jess rule 2
Upon event When condition Do action	(event) (not condition) => retract (event)	(event) (condition) => retract (event) (action)

Upon clause

- To implement event consumption
  - Events are always associated with a rule
- Event types as Jess Fact Templates

Event expression	Jess expression
e1&e2	(EventType1 (eventID “e1”) (ruleID “r1”)) (EventType2 (eventID “e2”) (ruleID “r1”))
{e1;e2} ! e3	(EventType1 (eventID “e1”) (ruleID “r1”) (timestamp ?t1)) (EventType2 (eventID “e2“) (ruleID “r1“) (timestamp ?t2&:(> ?t2 ?t1))) (not (EventType3 (eventID “e3“) (ruleID “r1“) (timestamp ?t3&:(and (>?t3 ?t1) (<?t3 ?t2)))))
e1|e2	(or (EventType1 (eventID “e1“) (ruleID “r1“)) (EventType2 (eventID “e2“) (ruleID “r1“)))
e1;e2	(EventType1 (eventID “e1“) (ruleID “r1“) (timestamp ?t1)) (EventType2 (eventID “e2“) (ruleID “r1“) (timestamp ?t2&:(> ?t2 ?t1)))



Upon clause (Detection Window)

```
(defrule DetectionWindowECAr1
(declare (salience 2))
?eventfact <-
(or (SituationContainedEvent (ruleID “r1”)
                                        (timestamp ?t&:(> (+ ?t dw) (call System currentTimeMillis))))
     (IncomeCall (ruleID “r1”)
                                         (timestamp ?t&:(> (+ ?t dw) (call System currentTimeMillis))))
     (SituationWithinRangeEvent (ruleID “r1”)
                                    (timestamp ?t&:(> (+ ?t dw) (call System currentTimeMillis)))))
=>
(retract ?eventfact))
```

Example (upon)

<u>ECA-DL:</u>

```
Upon EnterFalse (SituationContained (Person.John,
Building.HouseJohn)) ;
  EnterFalse (SituationContained (Person.Mary, Building.HouseJohn))
```

<u>JESS:</u>

```
(SituationContainedEvent (eventID “event1”) (ruleID “rule1”)
(containerID “HouseJohn”)
                                        (containedID “John”) (timestamp ?t1))
(SituationContainedEvent (eventID "event2") (ruleID "rule1")
(containerID “HouseJohn”)
                                        (containedID “Mary”)
(timestamp ?t2&:(> ?t2 ?t1)))
```

Example (ECA-DL)

```
Scope (EpilepticPatient.*; p) {
Upon EpilepticAlarm (p)
When p.hasHazardousActivity.hazardousvalue
Do NotifyPatientApplication (p) }
```

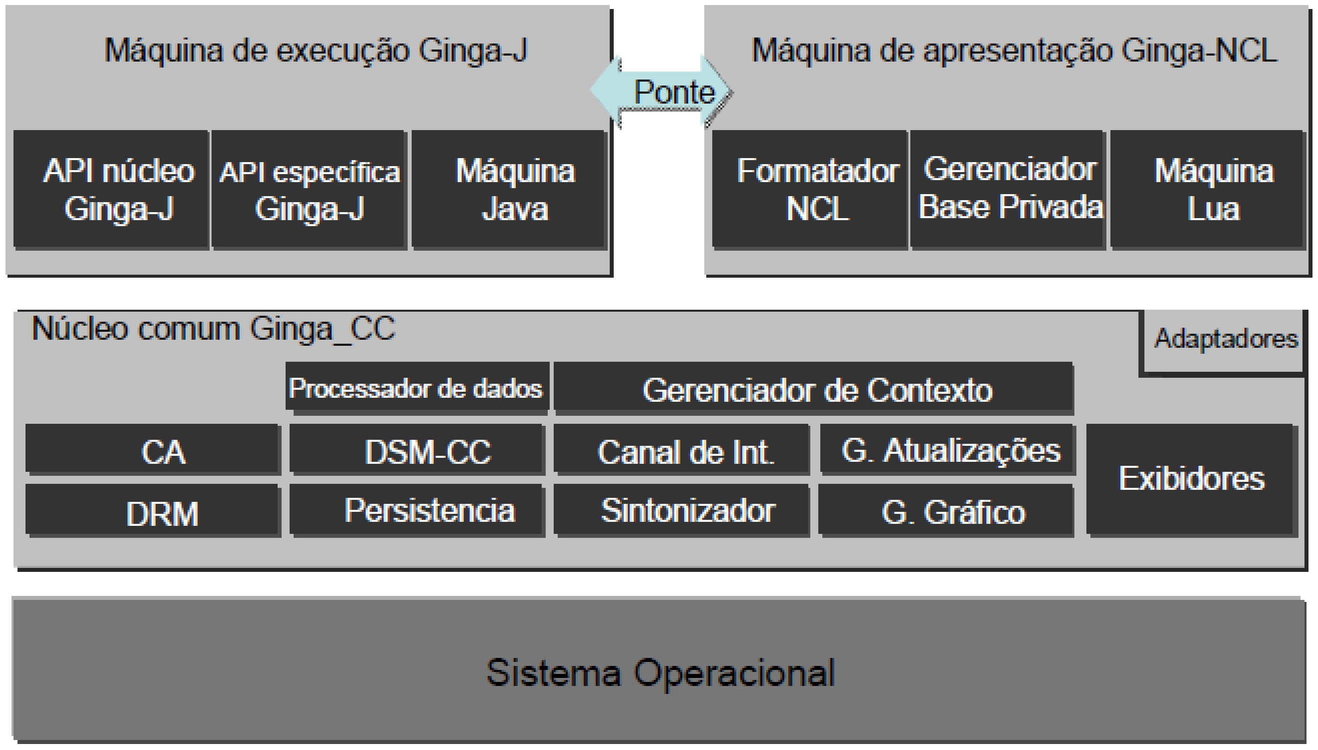
```
(defrule ECARule1_1
    ?event < -(EpilepticAlarm (idevent "ev1") (idrule "r11") (patientID ?patientId) (timestamp ?t))
    (EpilepticPatient (OBJECT ?patient)(identity ?patientId) (hazardousActivity ?ha))
    (not (HazardousActivity (OBJECT ?ha) (hazardousvalue ?value&:(=  ?value TRUE))))
    = >
    (retract ?event))

(defrule ECARule1_2
    ?event < -(EpilepticAlarm (idevent "ev1") (idrule "r11") (patientID ?patientId) (timestamp ?t))
    (EpilepticPatient (OBJECT ?patient) (identity ?patientId) (hazardousActivity ?ha))
    (HazardousActivity (OBJECT ?ha) (hazardousvalue ?value&: (=  ?value TRUE)))
    = >
    (retract ?event)
    (call HealthcareActionResolver notifyPatientApplication ?patientId))
```

ECA-DL to NCL

The SBTVD

- Better quality of images/videos;
- Provides interactivity;
- Includes standards for data modulation, transmission, data encoding, and a **middleware (coined Ginga)**.

Middleware Ginga



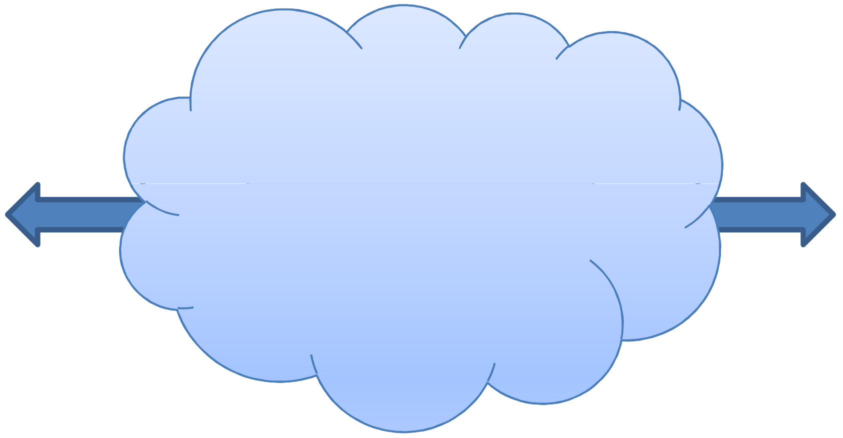
(SOARES e CASTRO, 2008)

Nested Context Language

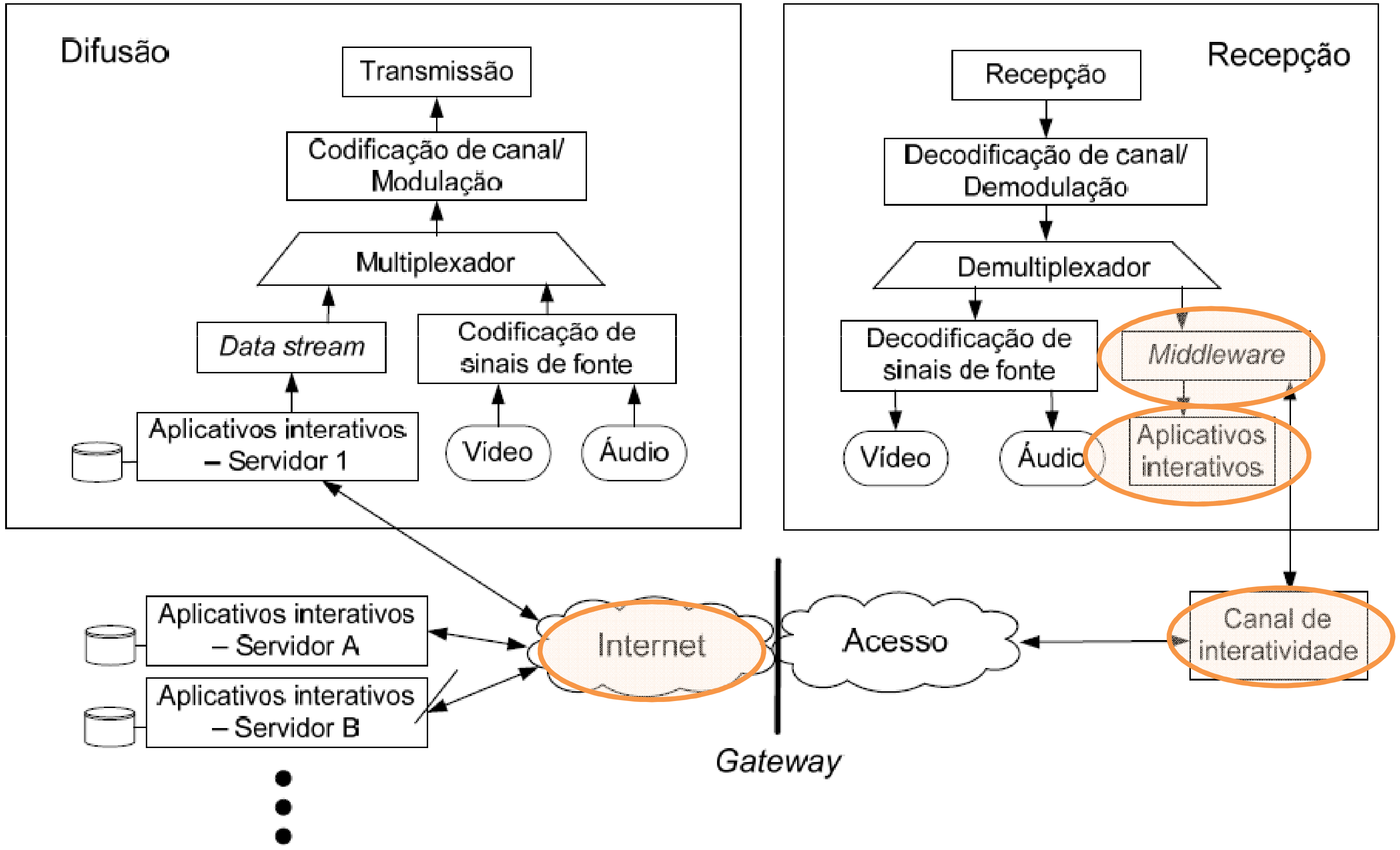
- Hypermedia authoring language used to describe multimedia applications with space-time synchronization between media objects (e.g., video, audio, images, etc.);
- As such, it does not have built-in concepts for context-handling and reactivity;
- NCLua
  - Scripting language;
  - Allows execution of imperative code in the declarative environment

Context in the SBTVD

Capturing Context

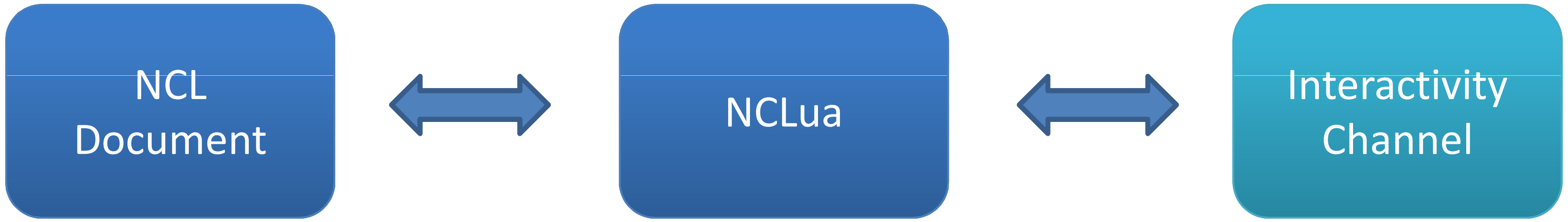


Interactivity Channel

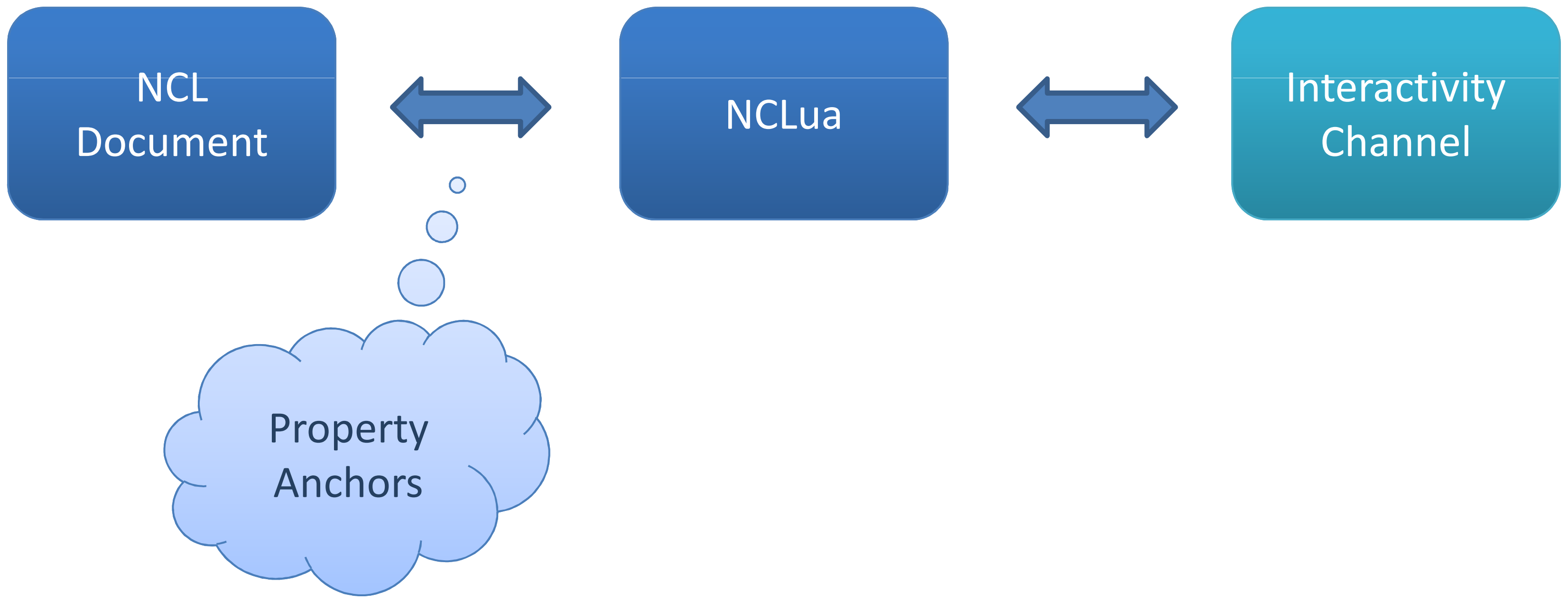

Difusão
Transmissão
Codificação de canal/ Modulação
Multiplexador
Data stream
Codificação de sinais de fonte
Aplicativos interativos – Servidor 1
Vídeo
Áudio
Recepção
Recepção
Decodificação de canal/ Demodulação
Demultiplexador
Decodificação de sinais de fonte
Middleware
Vídeo
Áudio
Aplicativos interativos
Aplicativos interativos – Servidor A
Aplicativos interativos – Servidor B
Internet
Acesso
Gateway
Canal de interatividade

(ABNT NBR15607-1, 2008)

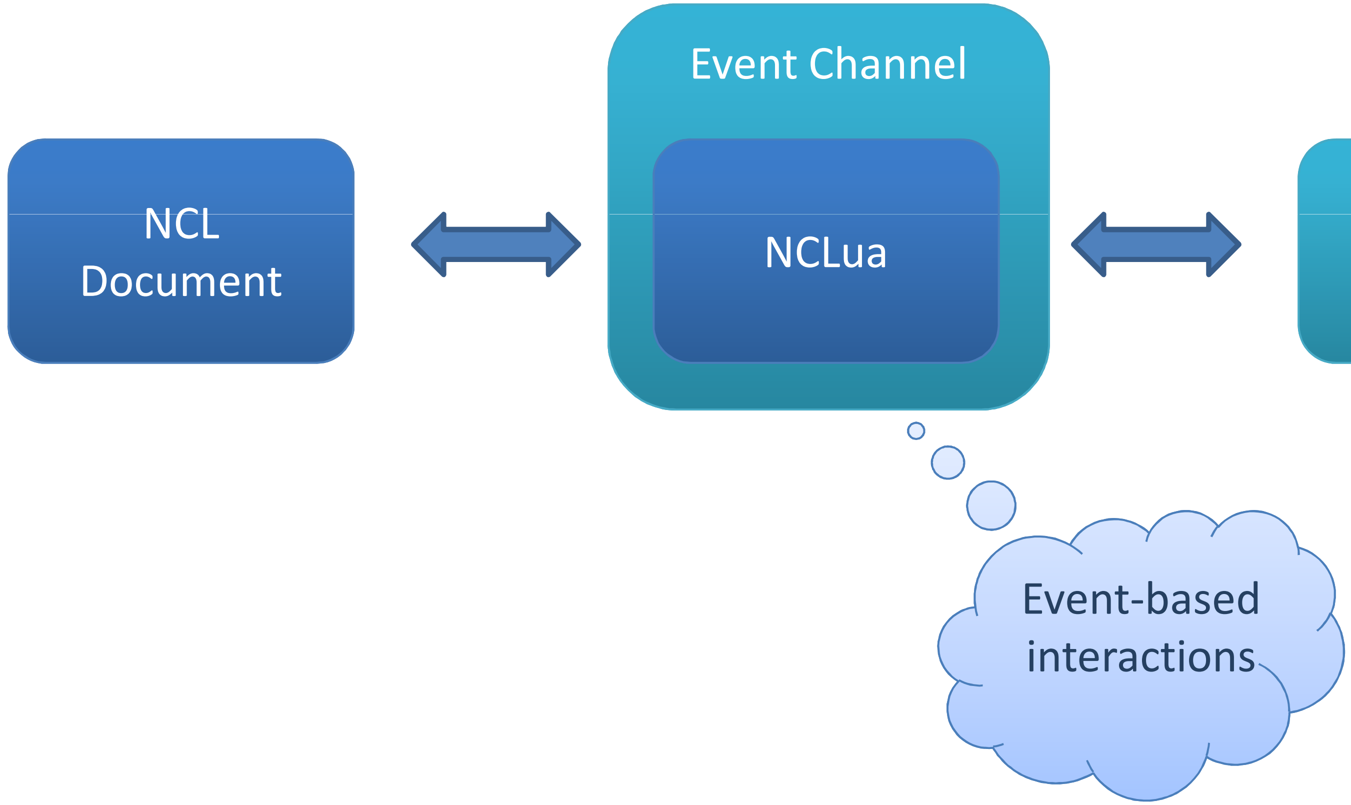
... in Ginga



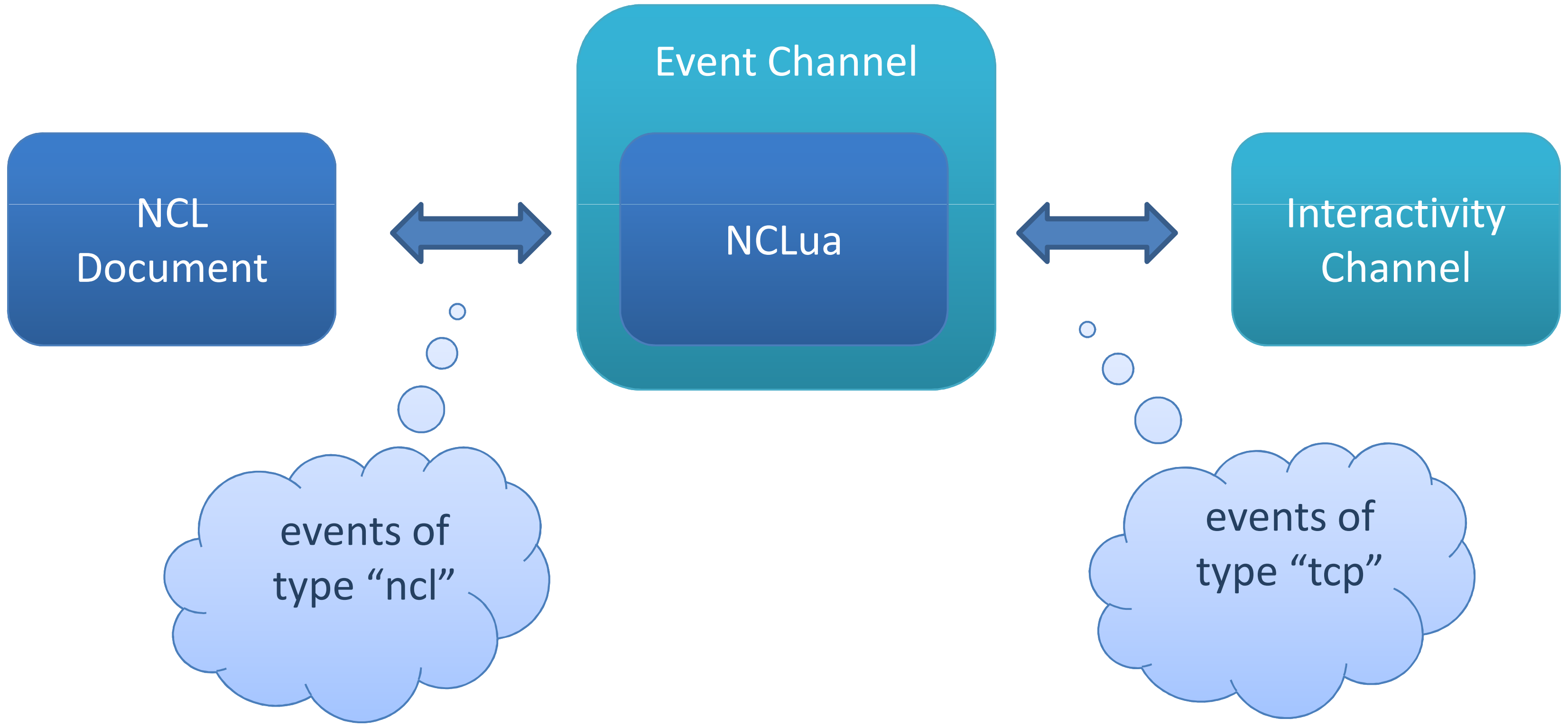
Representing Context



The Event Channel

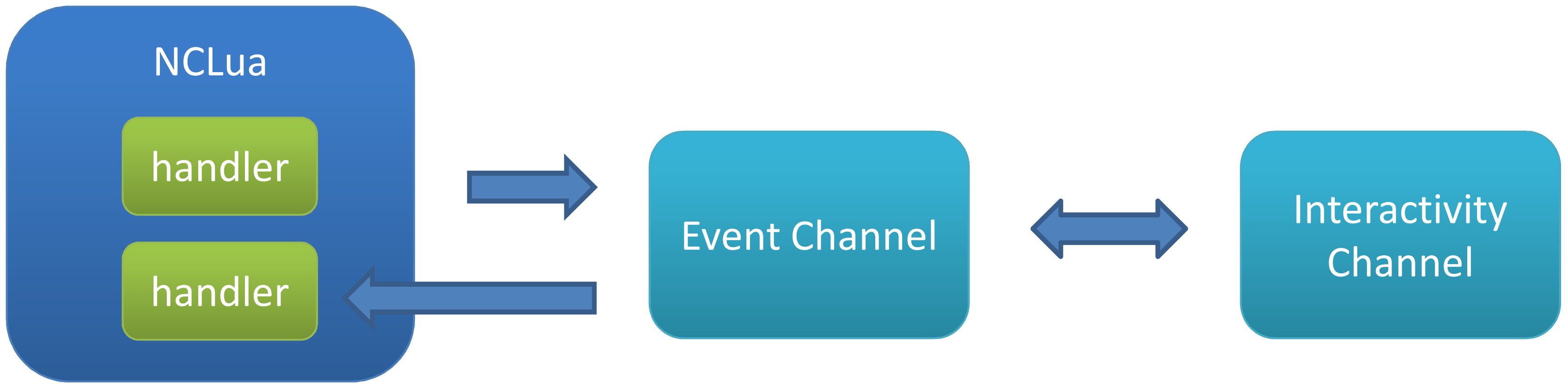


Events


NCL Document
Event Channel
NCLua
Interactivity Channel
events of type “ncl”
events of type “tcp”

Problems with the standard

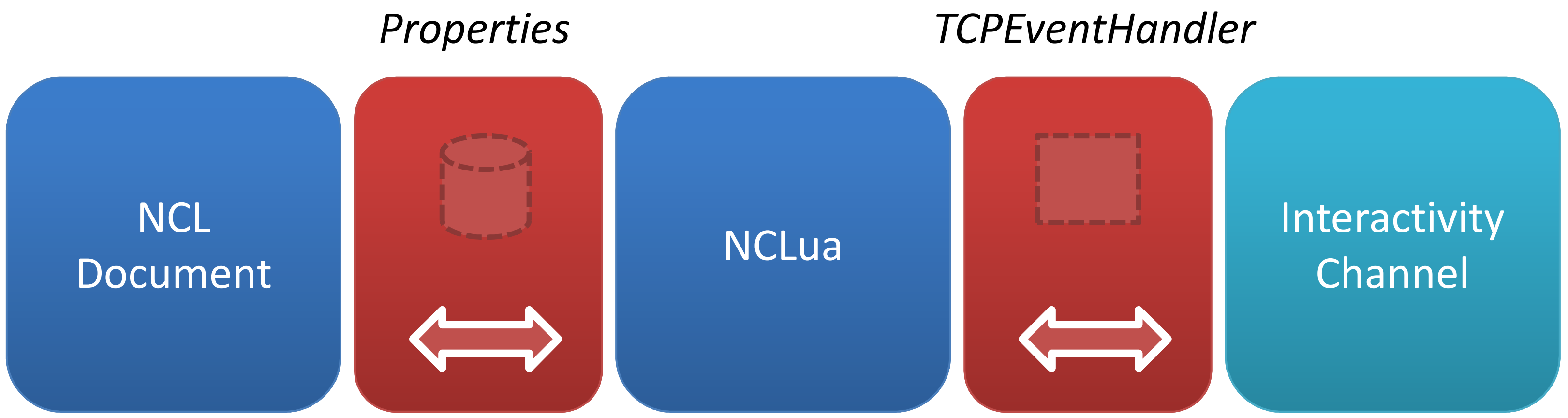
- It gets cumbersome when handling various context sources and context events



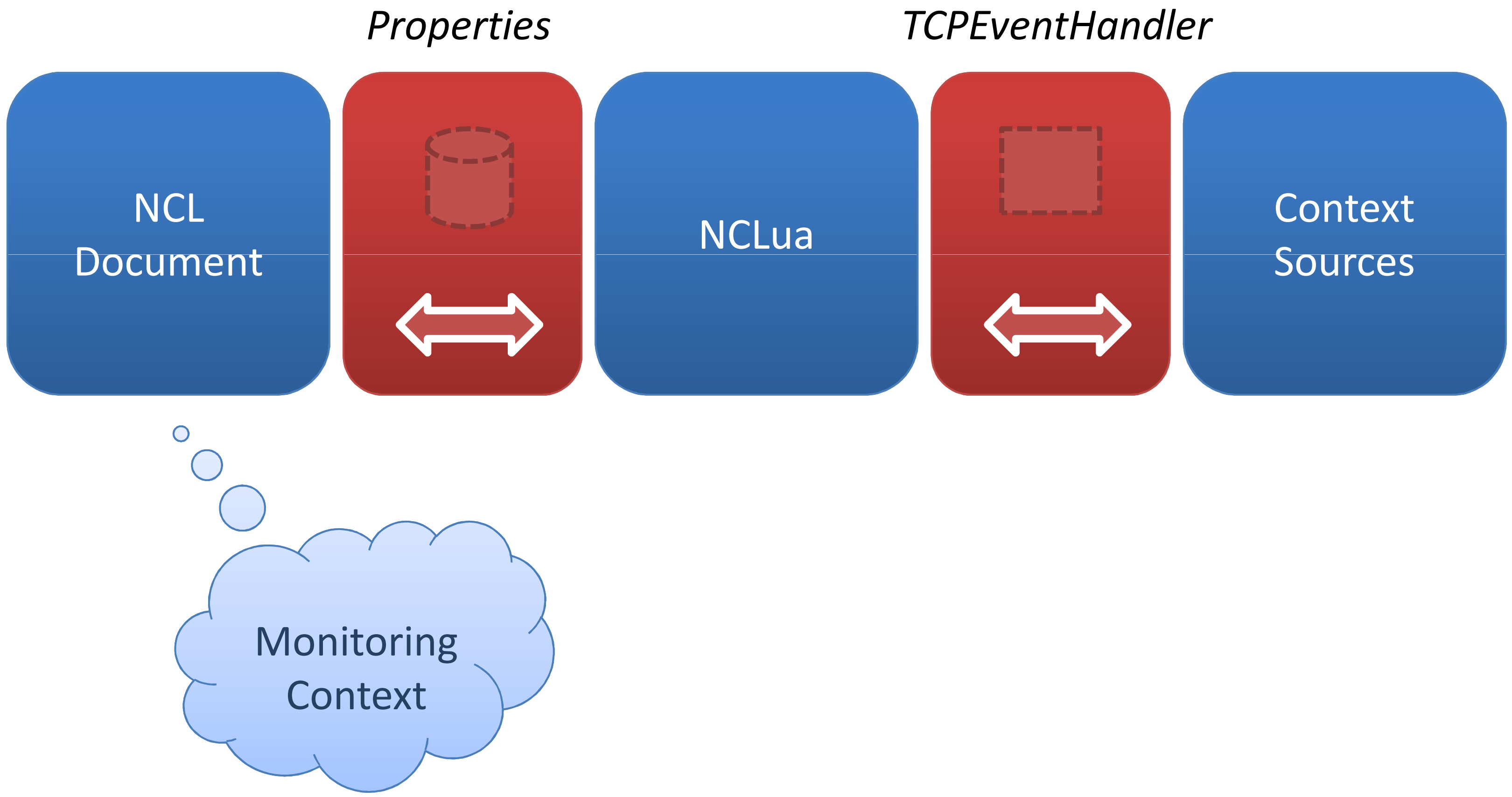
Contribution

- We have defined and implemented a **programming framework** that solves issues related to (context) event handling in the Ginga middleware;
  - **TCPEventHandler**: reusable code for facilitating the communication between NCL applications and external devices;
  - **Properties**: provides reusable code to help managing property anchor values in NCLua scripts
- Available online at [http://code.google.com/p/itveventframework/].

The Libraries



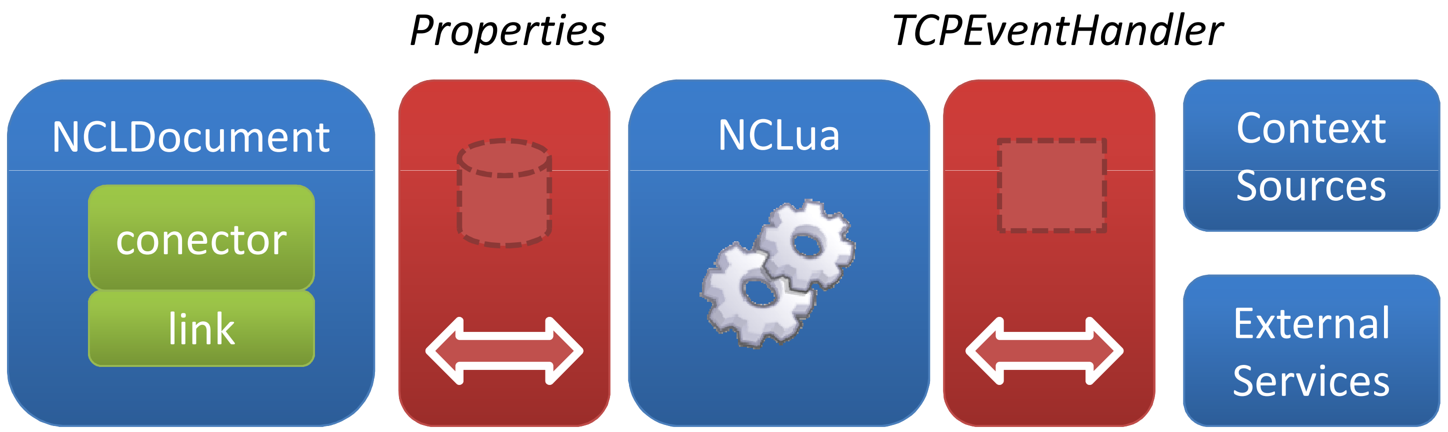
Reacting upon Context Changes



Connectors and Links

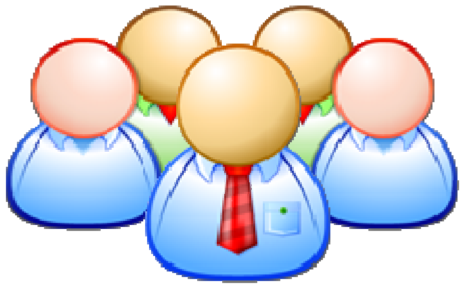
- Used to implement synchronism between NCL elements
  - E.g.: at the end of a video, starts another video
- Allows (with some adaptation) implementation of context-aware reaction rules

Now we know how to...



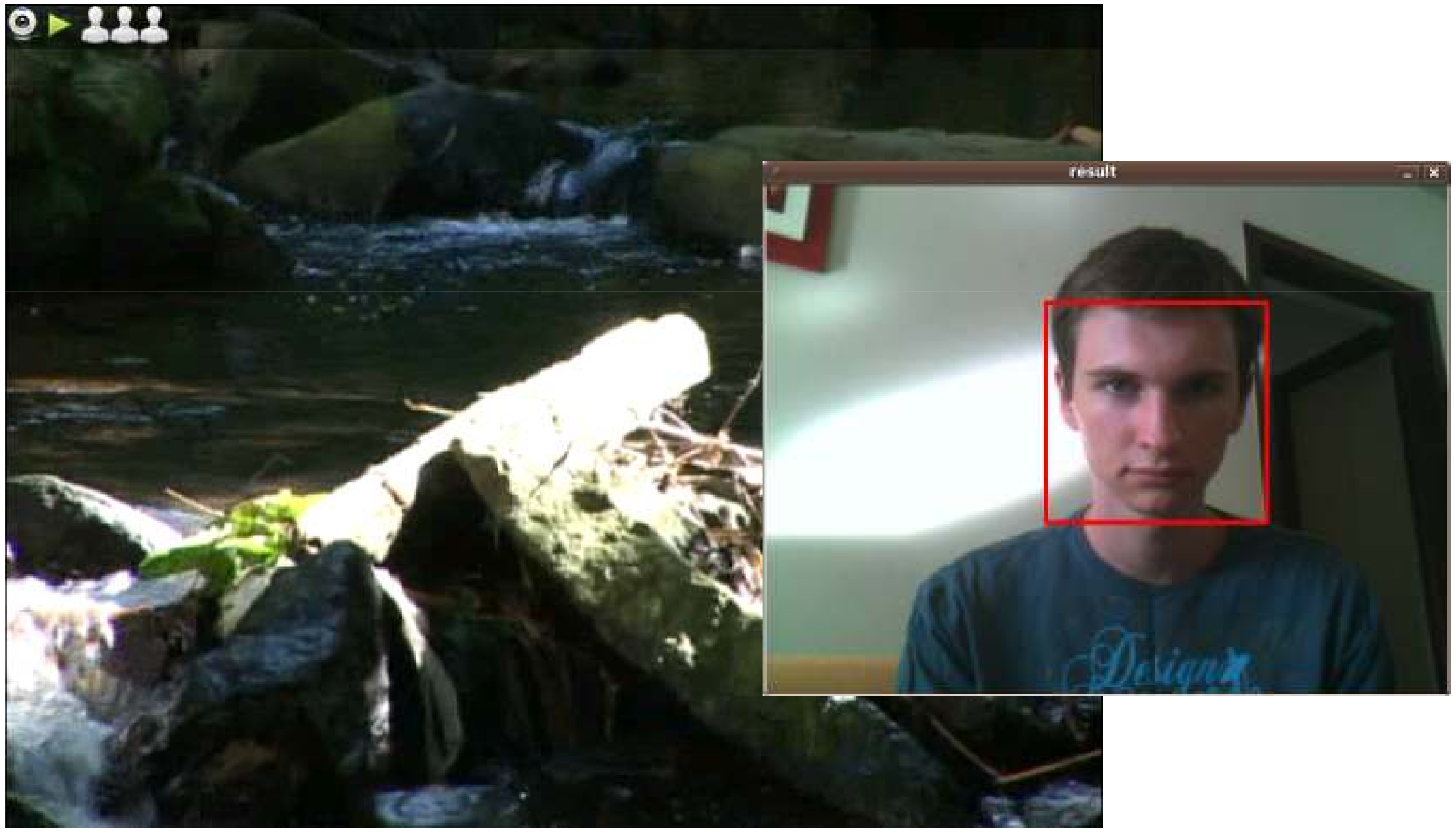
Prototypes

Audience Detector

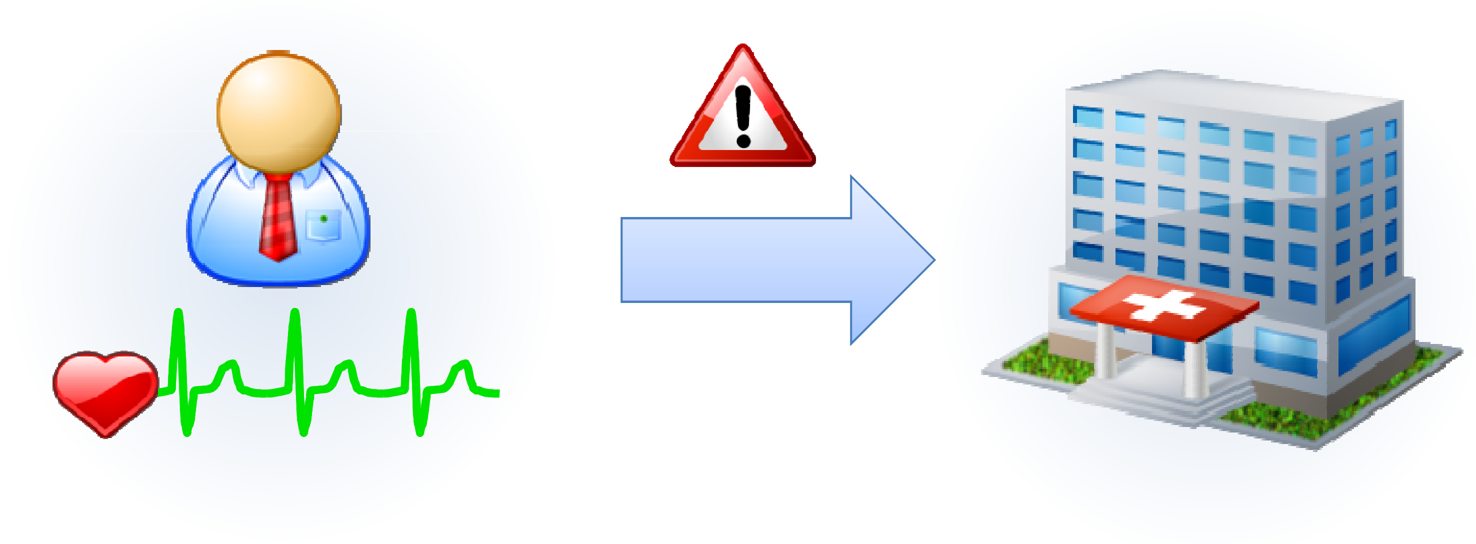


Audience Detector

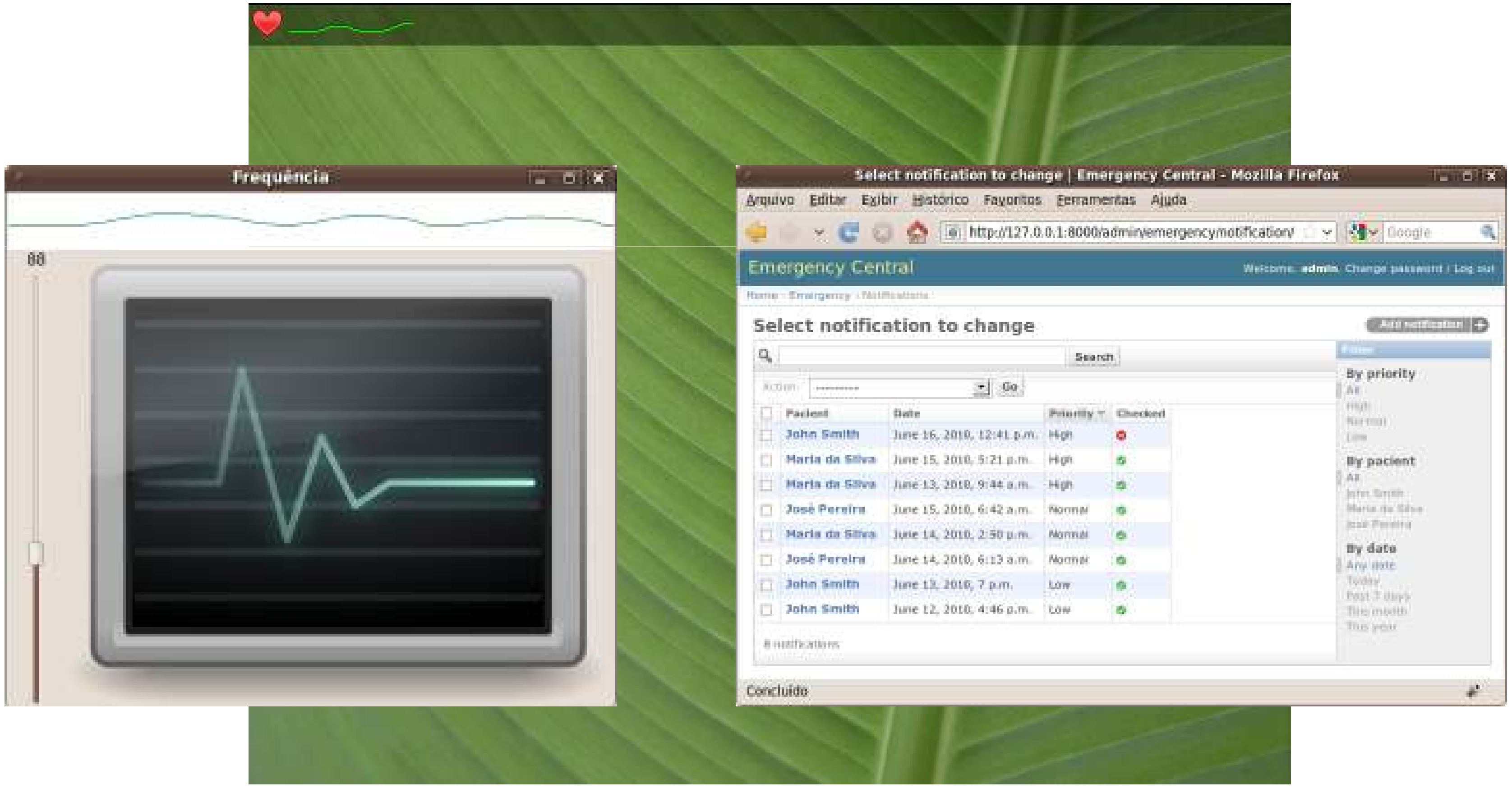
Execution examples



Heart Frequency Monitor



Heart Frequency Monitor


Frequência
88
Select notification to change | Emergency Central - Mozilla Firefox
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
http://127.0.0.1:8000/admin/emergency/notification/
Google
Emergency Central
Welcome, admin. Change password / Log out
Home › Emergency › Notifications
Add notification
Search
Action: ---------
Go
Filter
By priority
All
High
Normal
Low
By pacient
All
John Smith
Maria da Silva
José Pereira
By date
Any date
Today
Past 7 days
This month
This year
Concluído

Select notification to change

Pacient	Date	Priority	Checked
John Smith	June 16, 2010, 12:41 p.m.	High	🟥
Maria da Silva	June 15, 2010, 5:21 p.m.	High	🟩
Maria da Silva	June 13, 2010, 9:44 a.m.	High	🟩
José Pereira	June 15, 2010, 6:42 a.m.	Normal	🟩
Maria da Silva	June 14, 2010, 2:50 p.m.	Normal	🟩
José Pereira	June 14, 2010, 6:13 a.m.	Normal	🟩
John Smith	June 13, 2010, 7 p.m.	Low	🟩
John Smith	June 12, 2010, 4:46 p.m.	Low	🟩

8 notifications

Context and Situation Models, ECA-DL TVD and a Services Platform

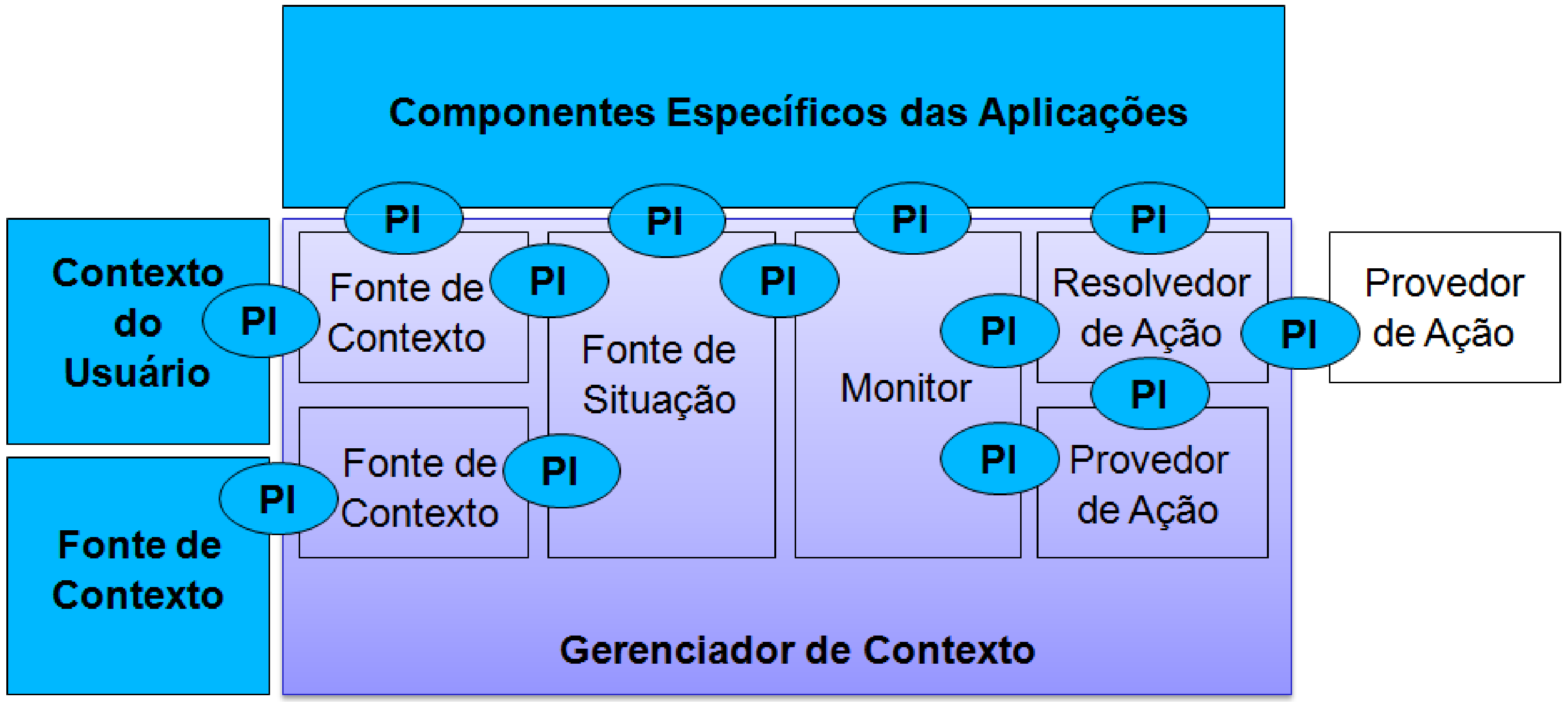
Motivation

- Ginga lacks generic services and there is no methodology for supporting context-aware development;
- The architecture should be in line with context models, situation models and ECA rules.

Requirements for a Context Management Platform

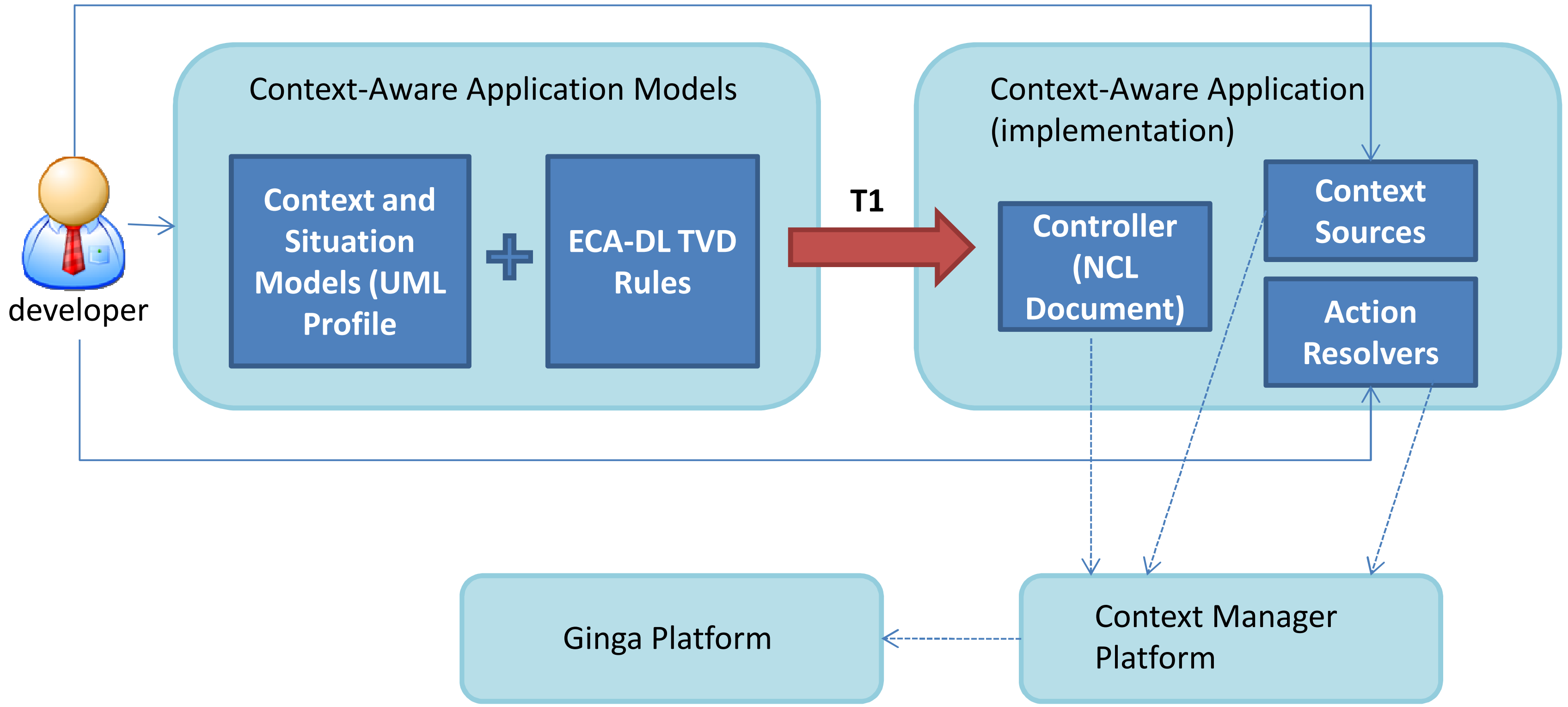
- Flexible and extensible;
- Perform context and situation reasoning;
- Support distribution;
- Mobile;
- Facilitate rapid development of context-aware applications.

The Context Management Platform Architecture

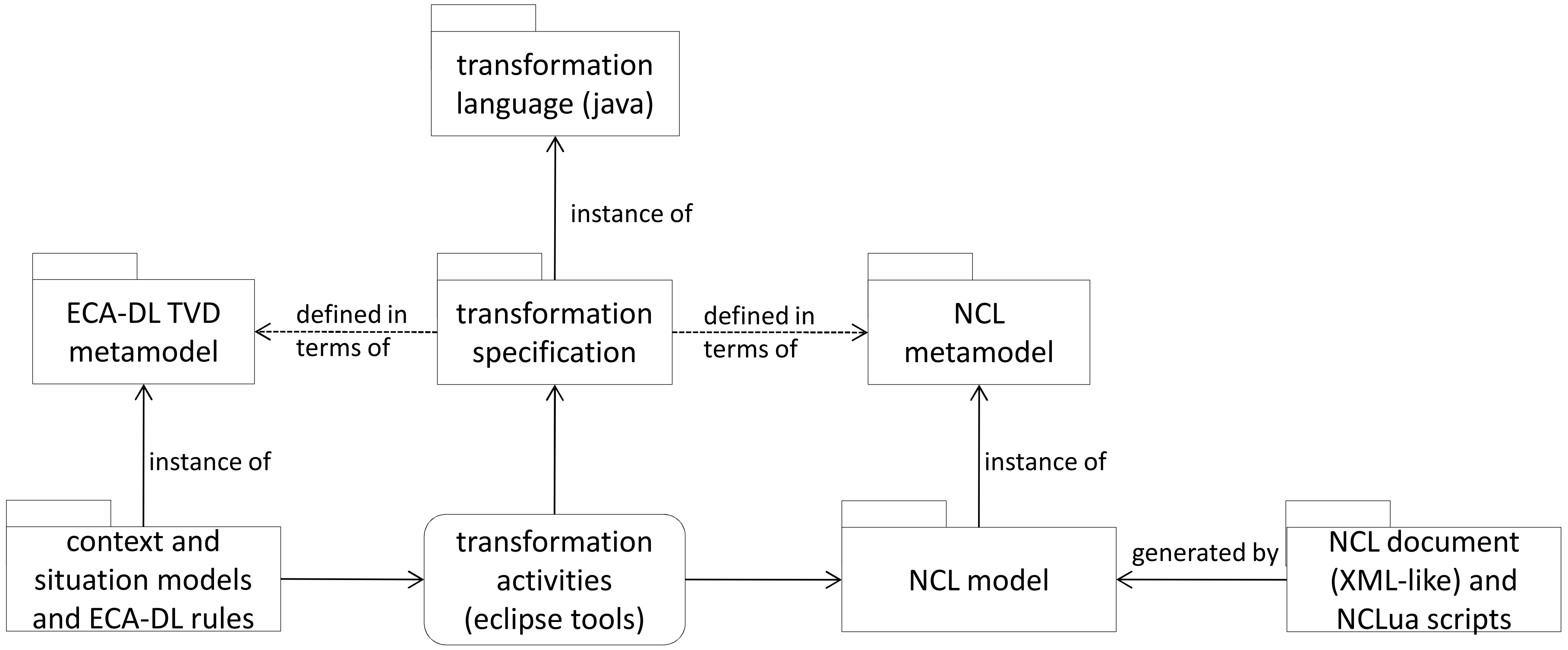


An MDA Approach

Overview


developer
Context-Aware Application Models
Context and Situation Models (UML Profile
+
ECA-DL TVD Rules
T1
Context-Aware Application (implementation)
Controller (NCL Document)
Context Sources
Action Resolvers
Ginga Platform
Context Manager Platform

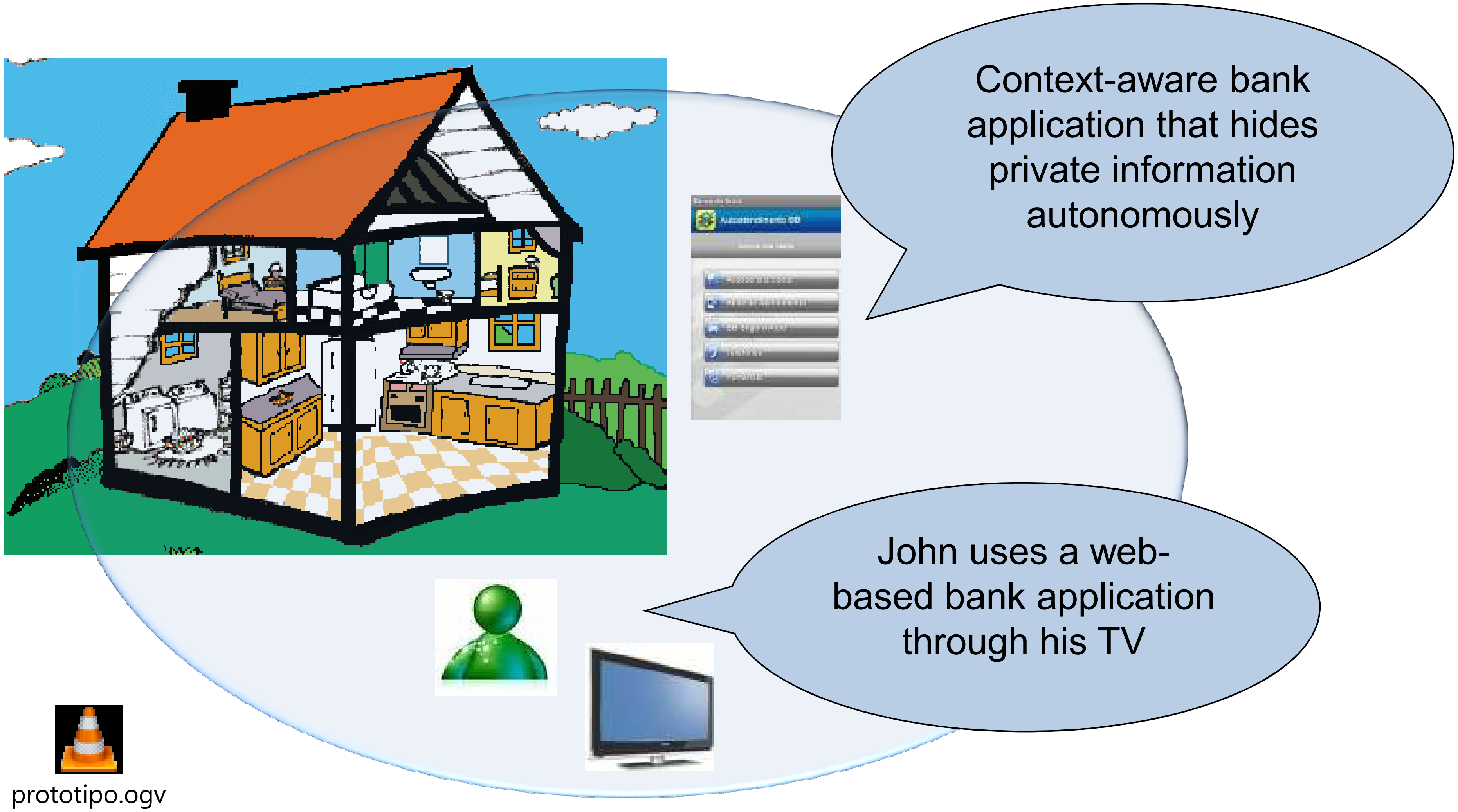
Transformation Overview


transformation language (java)
instance of
ECA-DL TVD metamodel
defined in terms of
transformation specification
defined in terms of
NCL metamodel
instance of
instance of
context and situation models and ECA-DL rules
transformation activities (eclipse tools)
NCL model
generated by
NCL document (XML-like) and NCLua scripts

Main Challenge of the Transformation

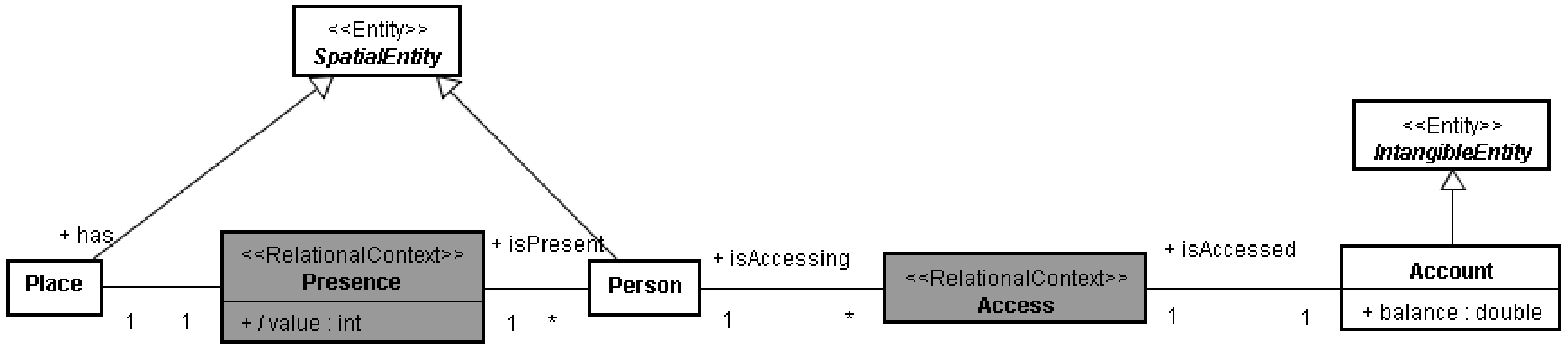
- Transforming a rule-based specification at the platform-independent level into an implementation platform that is not rule-oriented, while preserving conformance to the standard.

Case Study


Context-aware bank application that hides private information autonomously
John uses a web-based bank application through his TV

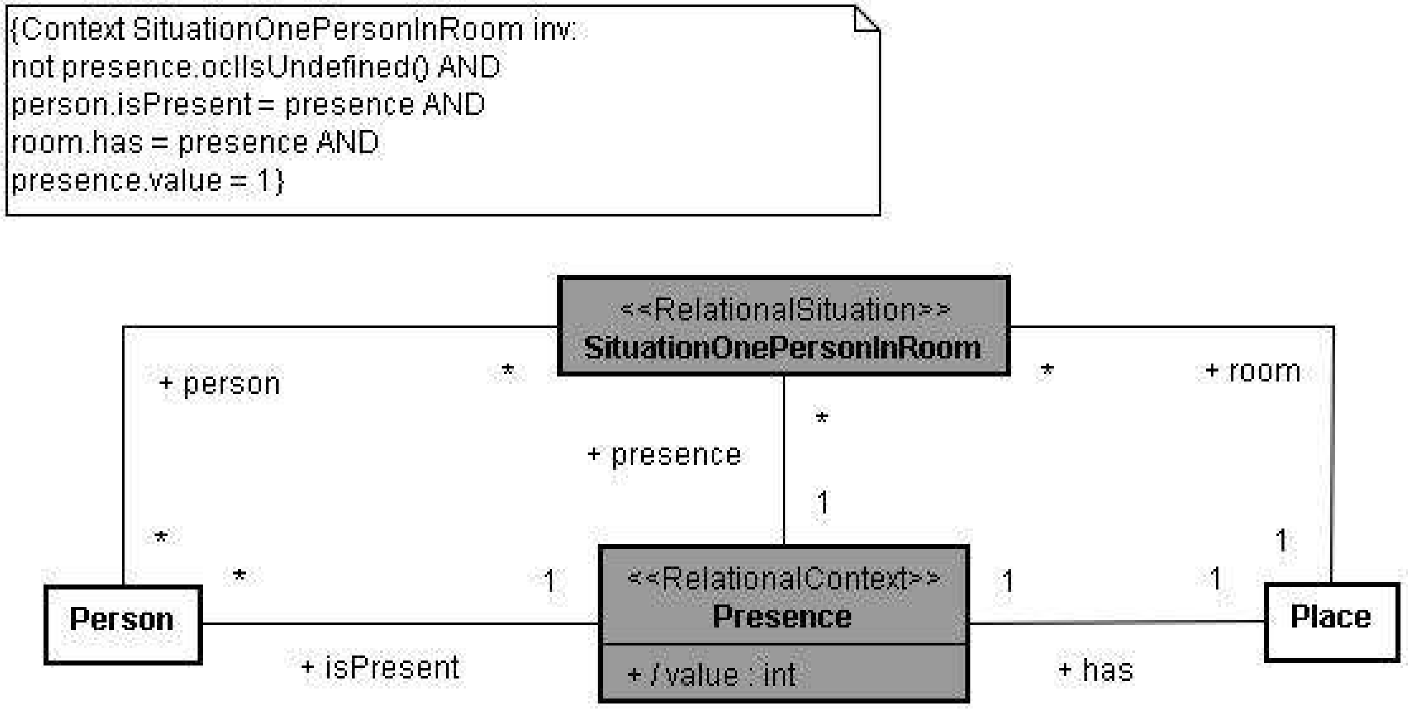
prototipo.ogv

Case Study: Context Model



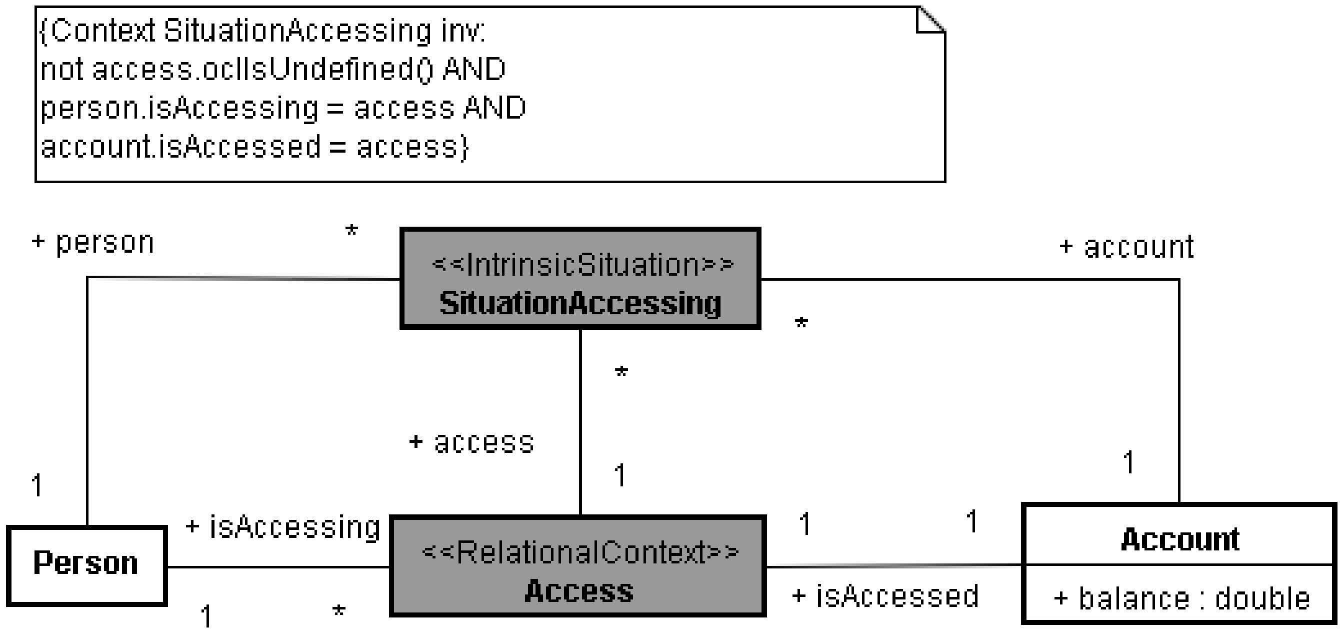
Case Study: Situation Model

- Situation “One person in the room”:


{Context SituationOnePersonInRoom inv:
not presence.oclIsUndefined() AND
person.isPresent = presence AND
room.has = presence AND
presence.value = 1}

Case Study: Situation Model

- Situation "Accessing":



ECA-DL TVD rules

Upon EnterTrue SituationAccessing (person1)
When SituationOnePersonInRoom (room1)
Do startWithBalance(account1)

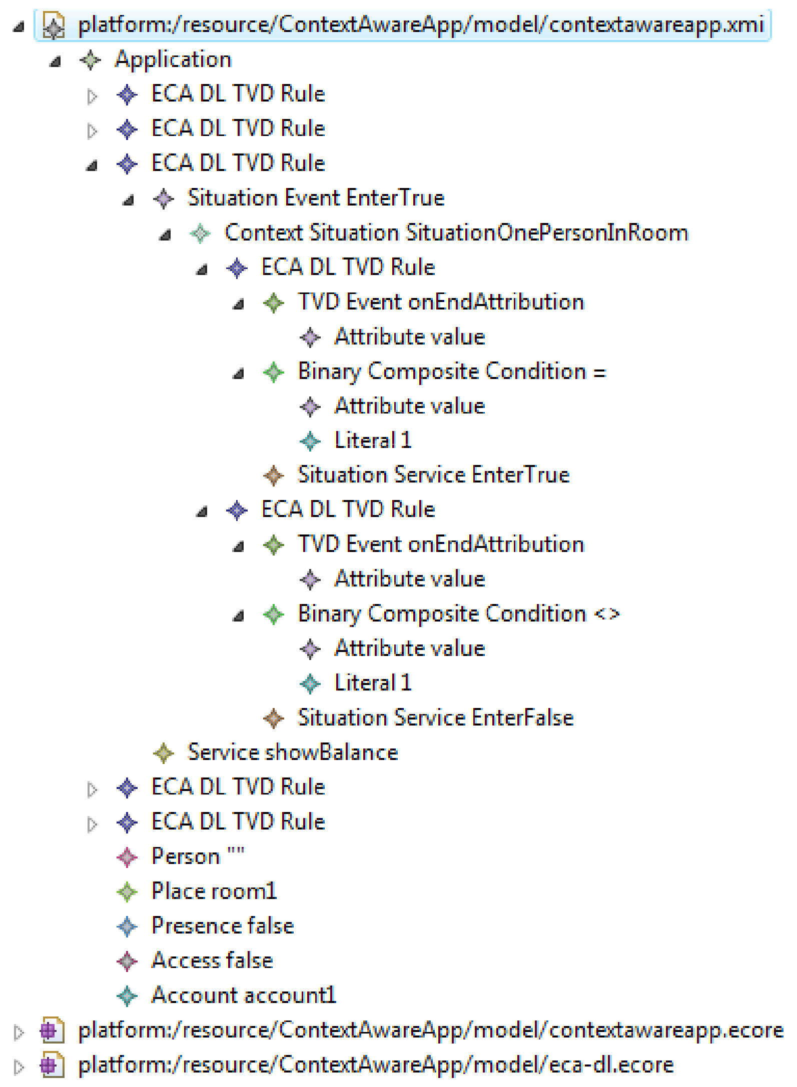
Upon EnterTrue SituationAccessing (person1)
When SituationMoreThanOnePersonInRoom (room1)
Do startWithoutBalance(account1)

Upon EnterTrue SituationOnePersonInRoom (room1)
Do showBalance(account1)

Upon EnterTrue SituationMoreThanOnePersonInRoom (room1)
Do hideBalance(account1)

Upon EnterTrue SituationNoPersonInRoom (room1)
Do logout(account1)

Ecore Model



```
◢ platform:/resource/ContextAwareApp/model/contextawareapp.xmi
  ◢ Application
    ▷ ECA DL TVD Rule
    ▷ ECA DL TVD Rule
    ◢ ECA DL TVD Rule
      ◢ Situation Event EnterTrue
        ◢ Context Situation SituationOnePersonInRoom
          ◢ ECA DL TVD Rule
            ◢ TVD Event onEndAttribution
                Attribute value
            ◢ Binary Composite Condition =
                Attribute value
                Literal 1
              Situation Service EnterTrue
          ◢ ECA DL TVD Rule
            ◢ TVD Event onEndAttribution
                Attribute value
            ◢ Binary Composite Condition <>
                Attribute value
                Literal 1
              Situation Service EnterFalse
        Service showBalance
    ▷ ECA DL TVD Rule
    ▷ ECA DL TVD Rule
      Person ""
      Place room1
      Presence false
      Access false
      Account account1
▷ platform:/resource/ContextAwareApp/model/contextawareapp.ecore
▷ platform:/resource/ContextAwareApp/model/eca-dl.ecore
```

Generated NCL Document

- Entities and Intrinsic Context types:

```
…
<media id="person1" src="scripts/person1.lua" descriptor="person1Desc">
         <property name="name" value=" " />
</media>
…
```

- Relational Context types:

```
…
<media id="Presence_person1_room1" src="scripts/Presence_person1_room1.lua"
descriptor="Presence_person1_room1_Desc">
         <property name="person" value="person1" />
         <property name="place" value="room1" />
         <property name="value" value=" " />
         <property name="exists" value="false" />
</media>
…
```

Generated Context Situations

```
…
<causalConnector id="SituationMoreThanOnePersonInRoom_room1_EnterFalse">
          <compoundCondition operator="and">
                    <simpleCondition role="upon10" transition="stops" eventType="attribution"
/>
                    <assessmentStatement comparator="lte">
                              <attributeAssessment role="when8" attributeType="nodeProperty"
eventType="attribution" />
                                        <valueAssessment value="1"/>
                              </assessmentStatement>
          </compoundCondition>
          <simpleAction role="action11" actionType="set" eventType="attribution" value="true"
/>
</causalConnector>
…
<link xconnector="SituationMoreThanOnePersonInRoom_room1_EnterFalse">
          <bind component="Presence_person1_room1" interface="value" role="upon10" />
          <bind component="Presence_person1_room1" interface="value" role="when8" />
          <bind component="SituationMoreThanOnePersonInRoom_room1" interface="end"
role="action11" />
</link>
…
```

Generated ECA-DL TVD

```
…
<causalConnector id="eca_dl_tvd_rule3">
          <compoundCondition operator="and">
                    <simpleCondition role="upon11" />
                    <assessmentStatement comparator="eq">
                              <attributeAssessment role="upon12" attributeType="nodeProperty"
eventType="attribution" />
                              <valueAssessment value="true"/>
                    </assessmentStatement>
          </compoundCondition>
          <simpleAction role="action14" actionType="set" eventType="attribution" value="true" />
</causalConnector>
…
<link xconnector="eca_dl_tvd_rule3">
          <bind component="SituationOnePersonInRoom_room1" interface="start" role="upon11" />
          <bind component="SituationOnePersonInRoom_room1" interface="start" role="upon12" />
          <bind component="account1" interface="showBalance" role="action14"/>
</link>
…
```

Contributions to the SBTVD

- Quite a few running context-aware applications;
- A programming framework;
- A methodology for developing context-aware applications, including:
  - modelling artefacts (context, situation and ECA-DL TVD)
  - a reference architecture;
  - a model-driven approach to generate NCL documents (+ NCL scripts) from high-level context-aware application specifications.

ECA-DL to Drools

Upon Clause with primitive events

Upon IncomeCall (Person.John, Person.Mary)

Do StartMonitoring()

```
declare IncomeCall
   @role(event)
end
```

```
rule ”rule 1”
     when
          $event: IncomeCall(entityFrom == “John”,  entityTo
     == “Mary”)
     then
          startMonitoring();
          retract($event);
end
```

Upon Clause with primitive events (cont.)

Upon IncomeCall(Person.Mary, Person.John) ;

IncomeCall(Person.Mary, Person.Fabio)

Do startMonitoring()

```
declare IncomeCall
   @role(event)
end
```

```
rule ”rule 1”
  when
       $event1: IncomeCall(entityFrom == “Mary”, entityTo ==
    “John”)
       $event2: IncomeCall(entityFrom == “Mary”, entityTo ==
    “Fabio”,  this after $event1)
  then
       startMonitoring();
       retract($event1);
       retract($event2);
  end
```

Upon Clause with primitive events and When Clause

Upon IncomeCall(Person.Mary, Person.John)

When Person.Mary.inRoom

Do startMonitoring()

```
rule ”rule 1”
when
//event
$event1: IncomeCall(entityFrom == “Mary”,
entityTo == “John”)
//condition
Person(name == “Mary”, inRoom == true)
then
    startMonitoring();
    retract($event1);
End
```

```
rule ”rule 1-2”
when
//event
$event1: IncomeCall(entityFrom == “Mary”,
entityTo == “John”)
//not condition
not( exists( Person(name == “Mary”, inRoom
    == true)))
then
    retract($event1);
end
```

Upon with EnterTrue() e EnterFalse() – Solution 1

Upon EnterTrue(Person.Mary.inRoom)

Do startWorking()

```
rule “Mary enters room”
  when
        Person(name == “Mary”, inRoom == true)
  then
        startWorking();
End
```

Upon with EnterTrue() e EnterFalse() – Solution 2

Upon EnterTrue(Person.Mary.inRoom)

Do startWorking()

```
declare
    EnterTruePerson
    Room
@role(event)
    person: Person
end
```

```
rule “rule 1”
when
$mary: Person(name == “Mary”,
    inRoom == true)
then
    insert(new
     EnterTruePersonRoom($mary));
end
```

```
rule “rule 2”
when
$e:
    EnterTruePersonRoom(person.
    name == “Mary”,
person.inRoom == true)
Then
     startWorking();
     retract($e);
End
```

Upon with EnterTrue() e EnterFalse() – Solution 3

Upon EnterTrue(Person.Mary.inRoom)

Do startWorking()

```
declare SituationFact
end
declare EnterTrueTransition
        @role(event)
         SituationFact: SituationFact
end
declare EnterFalseTransition
        @role(event)
        SituationFact: SituationFact
End
```

```
Declare PersonInRoom extends SituationFact
        person: Person
end
```

Upon EnterTrue(Person.Mary.inRoom)
Do startWorking()

```
rule “rule 1”
when
   $mary: Person(name == “Mary”, iRoom ==
      true)
   not (exists (PersonInRoom(person == $mary)))
then
        PersonInRoom spir = new
                       PersonInRoom($mary);
        insert(spir);
        insert(new
                EnterTrueTransition
                       ((SituationFact)spir));
end
```

```
rule “rule 2”
when
not (exists (Person(name == “Mary”, iRoom ==
       true)))
$spir: PersonInRoom(person.name == “Mary”,
       person.inRoom == true)
then
       retract($spir);
//inserir enterfalsetransition
end
```

Upon EnterTrue(Person.Mary.inRoom)

Do startWorking()

```
rule "rule 3 - evento EnterTrue desejado ocorre"
when
    $situation: PersonInRoom(person.name == "Mary",
    person.inRoom==true)
    $e: EnterTrueSituation(situationFact == $situation)
then
     startWorking();
     retract($e);
end
```

Upon EnterTrue(Person.Pablo.inRoom) ;

EnterTrue (Person.Pablo.hasSkypeStatus == SkypeStatus.online)

```
declare PersonInRoom extends SituationFact
    person: Pessoa
end

declare SkypeStatusOnline extends SituationFact
    person: Pessoa
end
```

Upon EnterTrue(Person.Pablo.inRoom) ;

EnterTrue (Person.Pablo.hasSkypeStatus == SkypeStatus.online)

```
//Geramos 5 regras em Drools!

rule "rule 1 – person in room"
when
      $pablo: Pessoa(nome == " Pablo", inRoom == true)
      not (exists (PersonInRoom(person == $mary)))
then
       PersonInRoom situation = new PersonInRoom($pablo);
       insert(situation);
       insert(new EnterTrueSituation ((SituationFact)situation));
end

// Regra para gerar EnterFalse, mais duas regras para SkypeStatus....
```

Upon EnterTrue(Person.Pablo.inRoom) ; EnterTrue (Person.Pablo.hasSkypeStatus == SkypeStatus.online)

```
rule “rule 5”
when
     $sitPablo:  PersonInRoom( person.nome == "Pablo", person.inRoom == true)
     $sitSkypeOnLine: SkypeStatus( person.nome == "Pablo", person.hasSkypeStatus ==
      SkypeStatus.online)
    //events
   $e1: EnterTrueSituation (situationFact == $sitPablo)
   $e2: EnterTrueSituationS (situationSKS == $sitSkypeOnLine, this after $e1)
then
   <...>
    retract($e1);
    retract($e2);
end
```

?