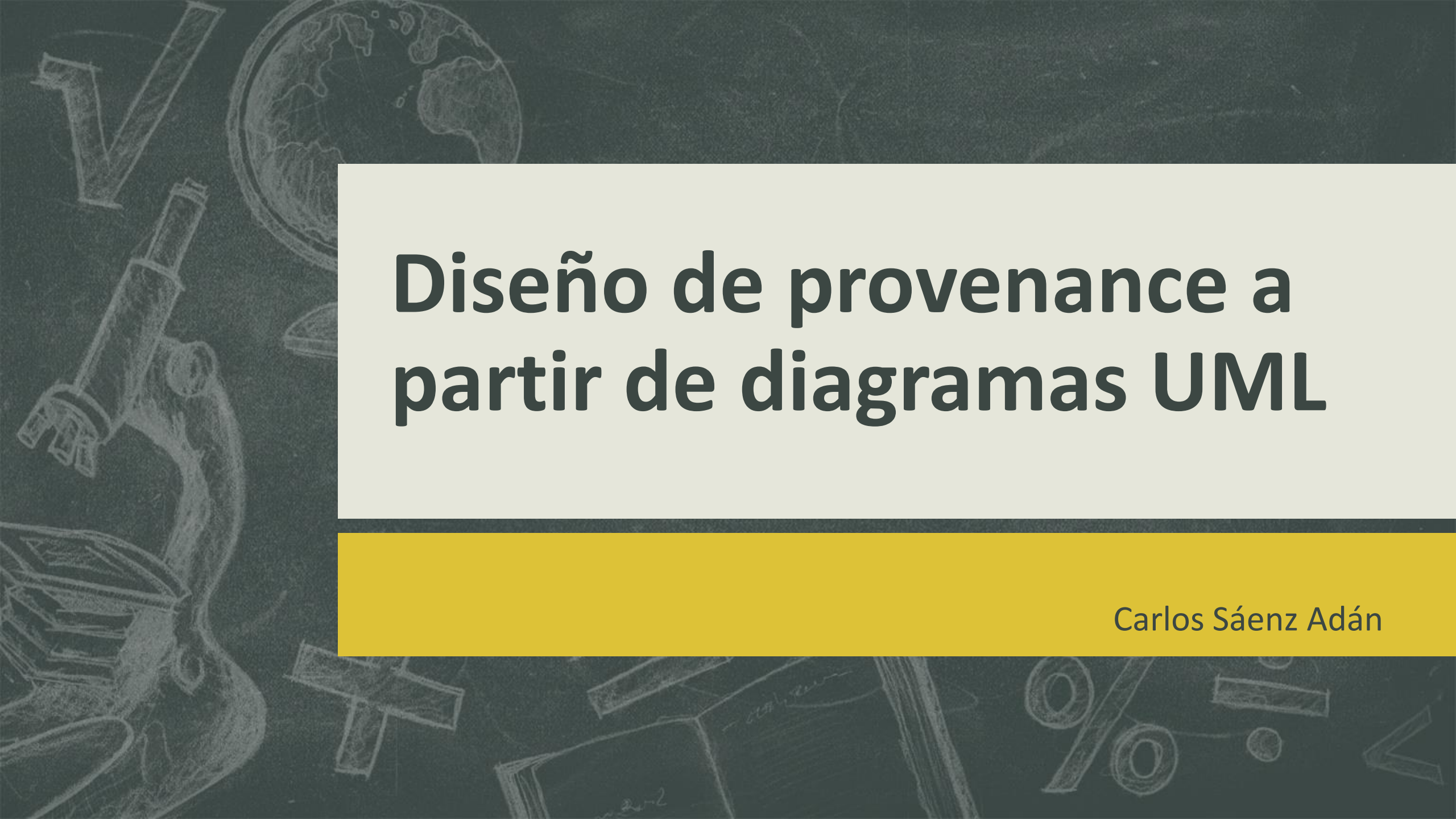


Diseño de provenance a partir de diagramas UML

Carlos Sáenz Adán

Índice

- Definición de *provenance*
- Ejemplos de *provenance*
- Objetivo
- W3C PROV standard
- PROV-Templates
- De UML a PROV
  - UML Sequence Diagrams
  - UML State Diagrams
- Extracción de Bindings
- Caso de estudio

Índice

- **Definición de *provenance***
- Ejemplos de *provenance*
- Objetivo
- W3C PROV standard
- PROV-Templates
- De UML a PROV
  - UML Sequence Diagrams
  - UML State Diagrams
- Extracción de Bindings
- Caso de estudio

Definición de *provenance*

Oxford English Dictionary

- (i) el hecho de proceder de un determinado **origen** o barrio; Origen, **derivación**.
- (ii) la **historia** o pedigrí de una obra de arte, manuscrito, libro raro, etc.; Concretamente, un registro de la **derivación** final y el paso de un artículo a través de sus diversos propietarios.

Merriam-Webster Online Dictionary

- (i) el **origen**, la **fuente**;
- (ii) la **historia** de la propiedad de un objeto valorado o una obra de arte o literatura.

Definición de *provenance*

procedencia

Der. del lat. *procēdens, -entis* 'procedente'.

1. f. Origen, principio de donde nace o se deriva algo.

2. f. Punto de partida de un barco, un tren, un avión, una persona, etc., cuando llega al término de su viaje.

3. f. Conformidad con la moral, la razón o el derecho.

4. f. *Der.* Fundamento legal y oportunidad de una demanda, petición o recurso.

indicación de procedencia

Definición de *provenance*

Moreau, L. (2010). The foundations for provenance on the web. *Foundations and Trends in Web Science*, *2*(2–3), 99-241.

Definition 3.3 (Provenance as Process) *The provenance of a piece of data is the process that led to that piece of data.* □

La procedencia de una pieza de datos es el proceso que condujo a esa pieza de datos

Definición de *provenance*

The W3C (World Wide Web Consortium) Provenance Working Group's definition of provenance:

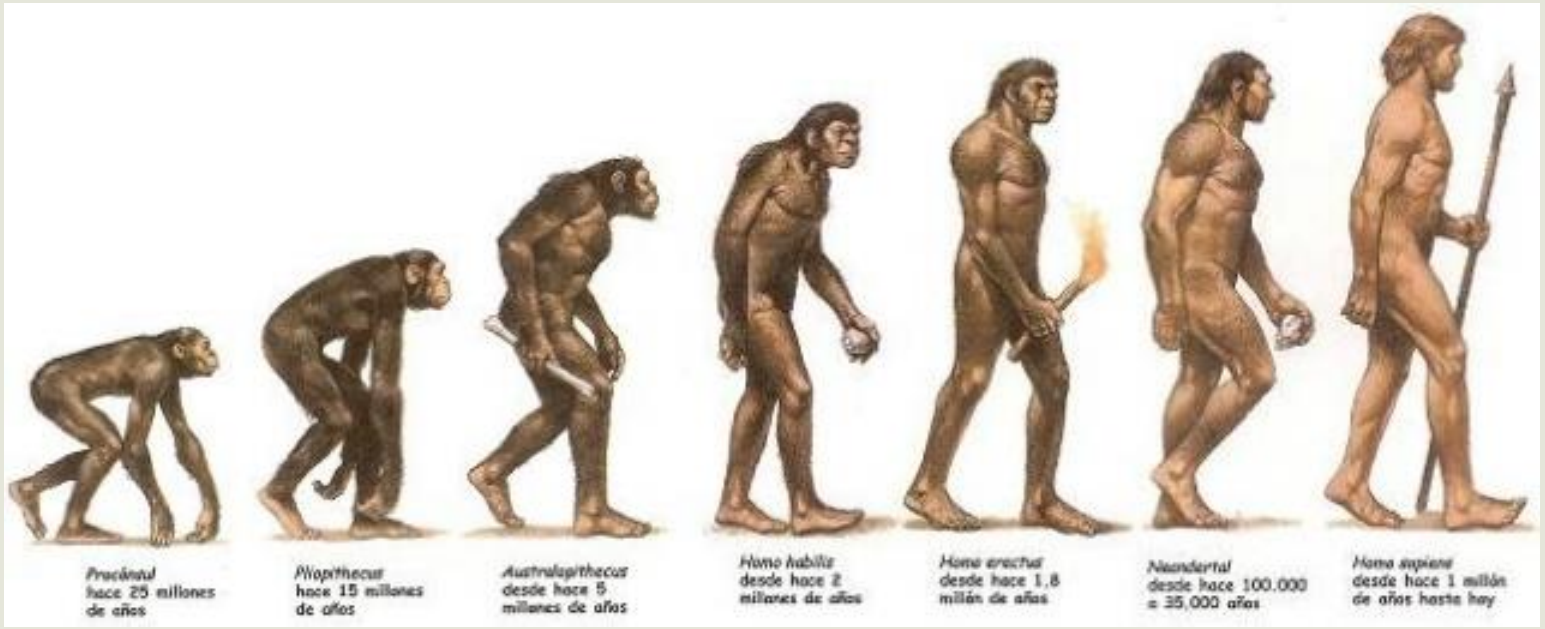
"Provenance is defined as a record that **describes** the people, institutions, entities, and activities **involved** in producing, influencing, or delivering a piece of data or a thing in the world"

"La procedencia se define como un registro que **describe** a las personas, instituciones, entidades y actividades **involucradas** con la producción, la influencia o la entrega de un dato o 'una cosa' en el mundo "

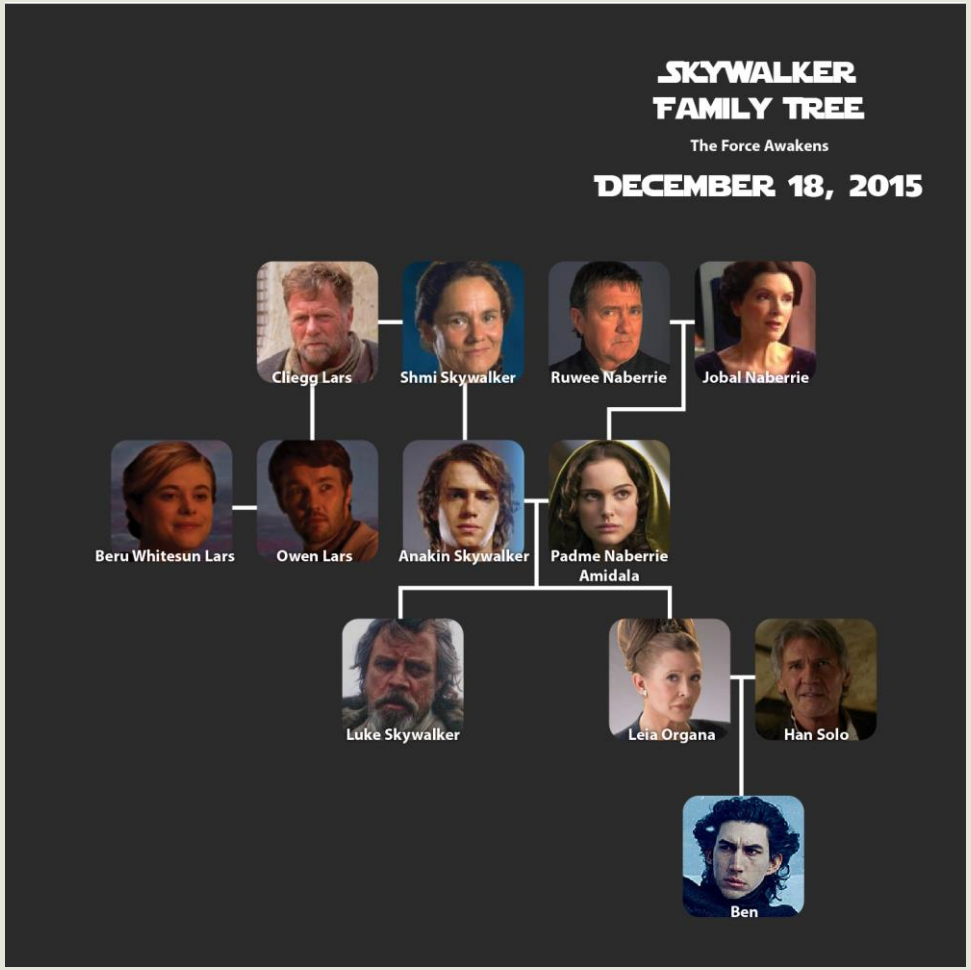
Índice

Ejemplo de *provenance*

La evolución

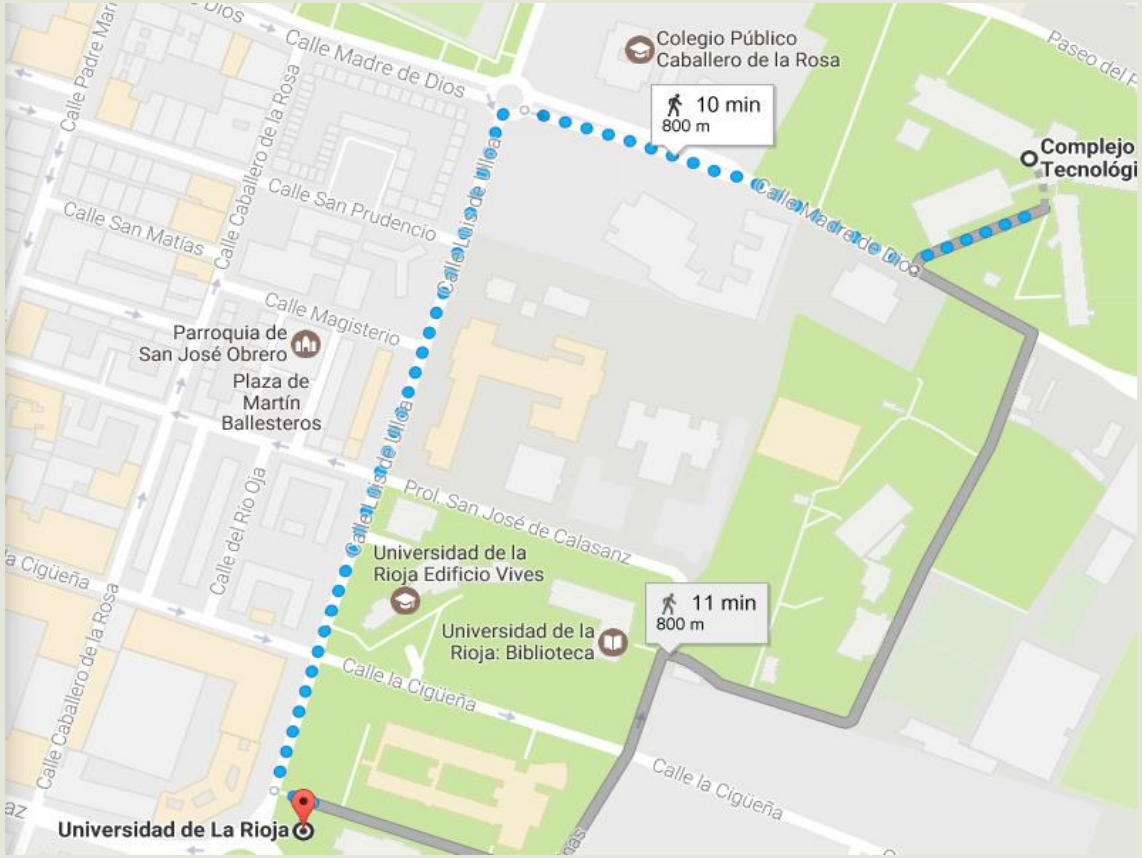


Ejemplo de *provenance*



Árbol genealógico

Ejemplo de *provenance*



Una ruta seguida

Ejemplo de *provenance*

StackTrace

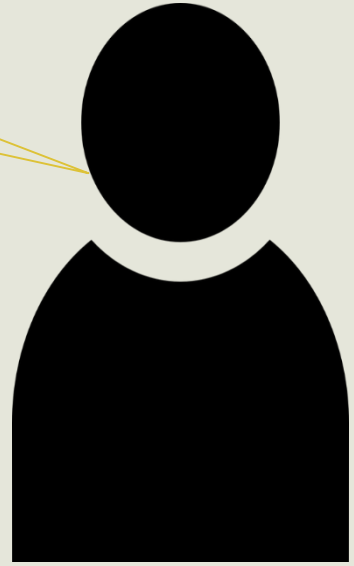
```
Exception in thread "main" java.lang.NullPointerException
        at templatematching.TestMatchWebCamArea.<init>(TestMatchWebCamArea.java:90)
        at templatematching.TestMatchWebCamArea.main(TestMatchWebCamArea.java:132)
```

Ejemplo de *provenance*

RIOJA | Denominación de Origen Calificada

Ejemplo de *provenance*

Tengo número de Erdös-Bacon-Sabbath (11)



Ejemplo de *provenance*

Tengo número de Erdös-Bacon-Sabbath (11)

¿Por qué? Demuéstralo

Ejemplo de *provenance*

Número de Erdös 4

MR Erdos Number = 3

Eladio Domínguez Murillo	coauthored with	Alberto Márquez Pérez	MR1151558
Alberto Márquez Pérez	coauthored with	Jaroslav Nešetřil	MR2547927
Jaroslav Nešetřil	coauthored with	Paul Erdős[1]	MR0737014

Change First Author | Change Second Author | New Search

Número de Sabbath 4

Soy miembro del grupo Sindicato de Riesgos que fue telonero del grupo de Ejea Tako
Tako han sido telonero de Los Suaves
Los Suaves fueron teloneros de Los Ramones
Los Ramones compartieron escenario con Black Sabbath ¡en Atlanta en 1978

Número de Bacon 3

Participé como figurante en la serie “Réquiem por Granada” con Horst Buchholz

Horst Buchholz aparece en "Avalanche Express" (“El tren de los espías”) con Maximilian Schell

Maximilian Schell aparece en “Telling lies in América” (“Ídolos, mentiras y rock and roll”) con Kevin Bacon.

Índice

Objetivo de *provenance*

Accountability

"La información *accountable* significa que el **uso de la información debe ser transparente**, de tal forma que sea posible determinar si un uso en particular es apropiado bajo un conjunto determinado de reglas, y que el sistema **permite que individuos e instituciones sean responsables** por el uso indebido“ (Weitzner et al.)

“Provenance es clave para permitir *accountable systems*, ya que consiste en una representación explícita de los procesos pasados, lo que nos permite rastrear el origen de los datos, las acciones y las decisiones” (Weitzner et al.)

Objetivo de *provenance*

- Trazabilidad
- Integridad
- Responsabilidad
- Calidad de los resultados
- Validación de los resultados
- Reproducir resultados
- Encontrar errores

Objetivo de *provenance*

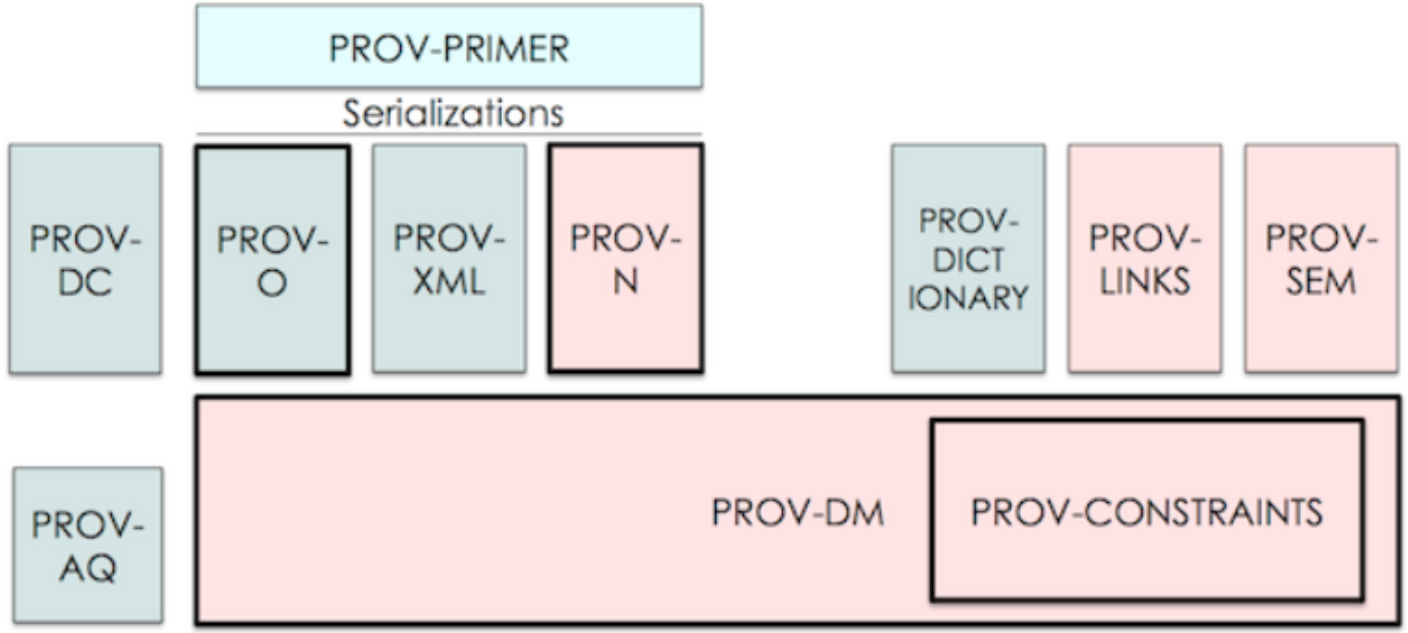
- Trazabilidad
- Integridad
- Responsabilidad
- Calidad de los resultados
- Validación de los resultados
- Reproducir resultados
- Encontrar errores

Confiar en los datos/resultado

Índice

- Definición de provenance
- Ejemplos de provenance
- Objetivo
- **W3C PROV standard**
- PROV-Templates
- De UML a PROV
  - UML Sequence Diagrams
  - UML State Diagrams
- Extracción de Bindings
- Caso de estudio

Provenance estándar. W3C PROV
Family of documents



Users. Quieren entender PROV y usar aplicaciones que soportan PROV

Developers. Quieren desarrollar aplicaciones que generan y consumen PROV

Advanced. Quieren crear validadores, nuevas formas de serialización, o sistemas avanzados de provenance.

https://www.w3.org/TR/prov-overview/

Provenance estándar. W3C PROV

Family of documents

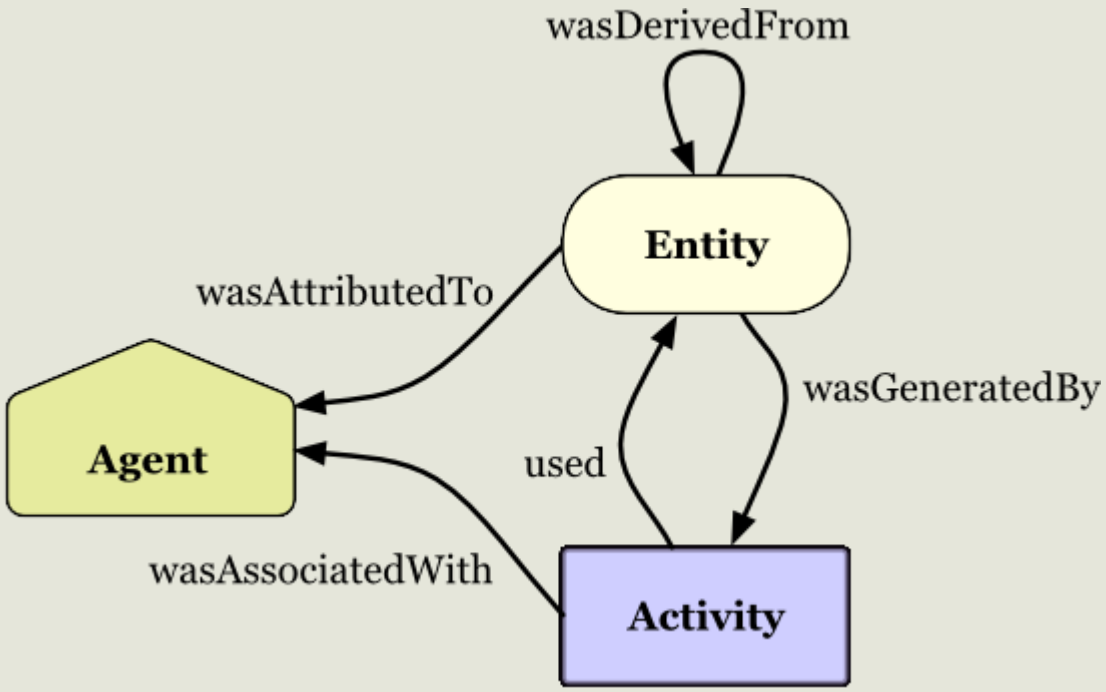
PROV-OVERVIEW. Descripción general de la familia de documentos PROV.
PROV-PRIMER. Manual básico del modelo de datos PROV.
PROV-O. Ontología de PROV en OWL2. Permite el mapeo entre PROV y RDF.
PROV-DM. El modelo de datos PROV para provenance.
PROV-N. Notación PROV legible para las personas.
PROV-CONSTRAINTS. Conjunto de restricciones aplicables al modelo de datos PROV.
PROV-XML. XML Schema del modelo de datos PROV
PROV-AQ. Mecanismo para el acceso y consulta de provenance.
PROV-DICTIONARY. Añade un nuevo tipo de colección.
PROV-DC. Mapping entre PROV-O y Dublin Core Terms.
PROV-SEM. Especificación en términos de lógica de primer orden.
PROV-LINKS. Mecanismo para relacionar bundles.

Provenance estándar. W3C PROV

Family of documents

PROV-OVERVIEW. Descripción general de la familia de documentos PROV.
PROV-PRIMER. Manual básico del modelo de datos PROV.
PROV-O. Ontología de PROV en OWL2. Permite el mapeo entre PROV y RDF.
PROV-DM. Especificación del modelo de datos PROV para *provenance*.
PROV-N. Notación PROV legible para las personas.
PROV-CONSTRAINTS. Conjunto de restricciones aplicables al modelo de datos PROV.
PROV-XML. XML Schema del modelo de datos PROV
PROV-AQ. Mecanismo para el acceso y consulta de provenance.
PROV-DICTIONARY. Añade un nuevo tipo de colección.
PROV-DC. Mapping entre PROV-O y Dublin Core Terms.
PROV-SEM. Especificación en términos de lógica de primer orden.
PROV-LINKS. Mecanismo para relacionar bundles.

W3C PROV Elements

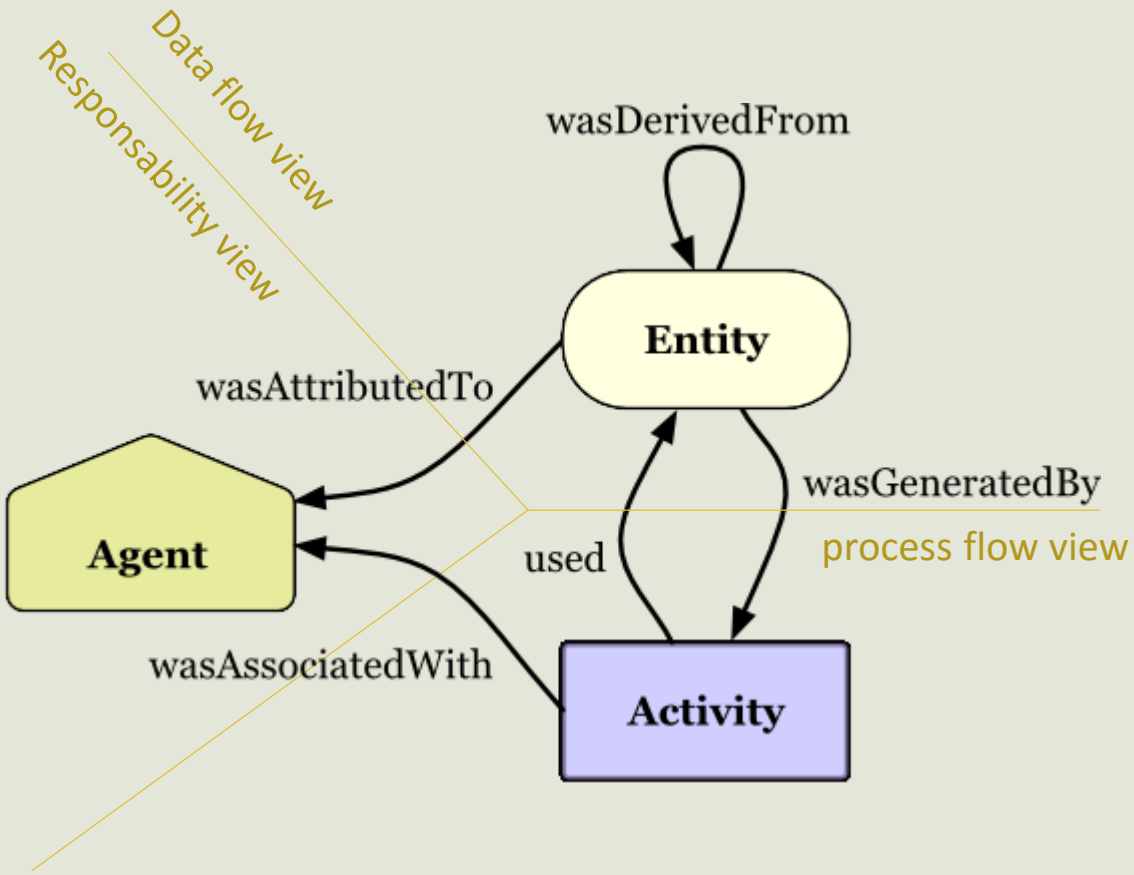


Entity: es algo físico, digital, conceptual o de otro tipo con algunas características fijadas. Puede ser real o imaginario.

Activity: algo que ocurre durante un periodo de tiempo y actúa sobre entidades, puede incluir el consume, procesamiento, transformación, modificación, uso, generación de entidades

Agent: algo que tiene algún tipo de responsabilidad sobre la ejecución de una actividad, la existencia de una entidad o la actividad de otro agente.

W3C PROV Elements



Entity: es algo físico, digital, conceptual o de otro tipo con algunas características fijadas. Puede ser real o imaginario.

Activity: algo que ocurre durante un periodo de tiempo y actúa sobre entidades, puede incluir el consume, procesamiento, transformación, modificación, uso, generación de entidades

Agent: algo que tiene algún tipo de responsabilidad sobre la ejecución de una actividad, la existencia de una entidad o la actividad de otro agente.

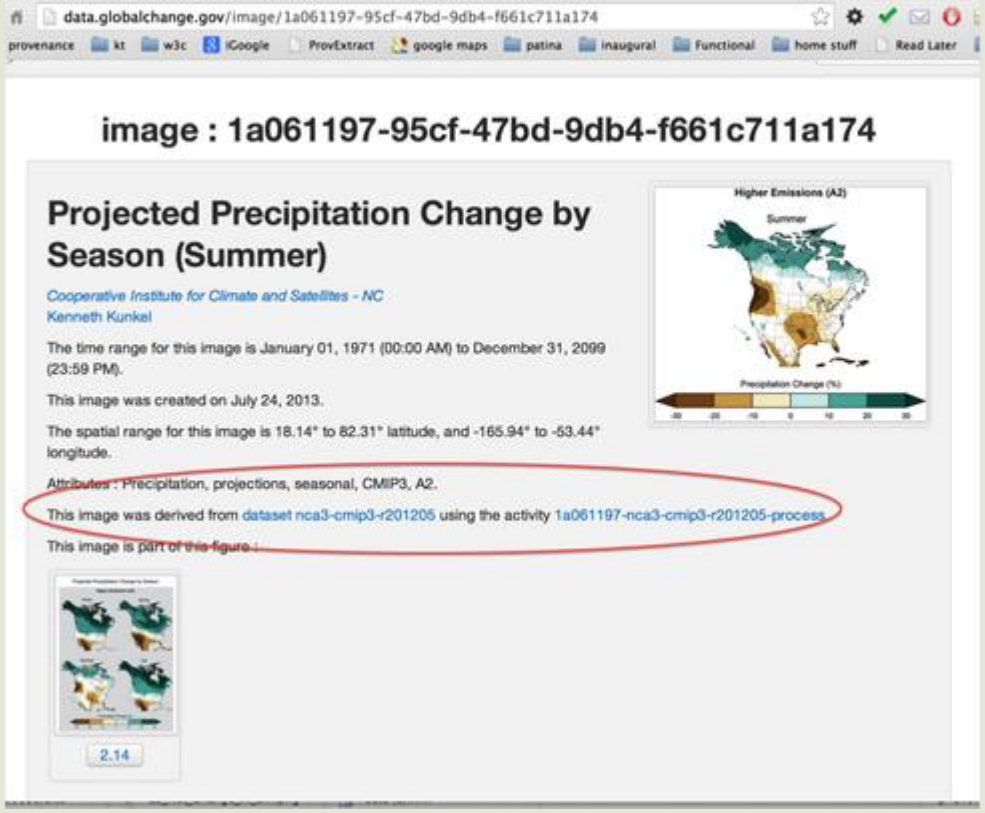
W3C PROV Elements

Type or Relation Name	Representation in the PROV-N notation	Component
Entity	**entity(id, [attr1=val1, ...])**	Component 1: Entities/Activities
Activity	**activity(id, st, et, [attr1=val1, ...])**	
Generation	**wasGeneratedBy(**id;**e,a**,t,attrs**)**	
Usage	**used(**id;**a,e**,t,attrs**)**	
Communication	**wasInformedBy(**id;**a2,a1**,attrs**)**	
Start	wasStartedBy(id;a2,e,a1,t,attrs)	
End	wasEndedBy(id;a2,e,a1,t,attrs)	
Invalidation	wasInvalidatedBy(id;e,a,t,attrs)	
Derivation	**wasDerivedFrom(**id; **e2, e1**, a, g2, u1, attrs**)**	Component 2: Derivations
Revision	... prov:type='prov:Revision' ...	
Quotation	... prov:type='prov:Quotation' ...	
Primary Source	... prov:type='prov:PrimarySource' ...	
Agent	**agent(id, [attr1=val1, ...])**	Component 3: Agents, Responsibility, Influence
Attribution	**wasAttributedTo(**id;**e,ag**,attr**)**	
Association	**wasAssociatedWith(**id;**a,ag**,pl,attrs**)**	
Delegation	**actedOnBehalfOf(**id;**ag2,ag1**,a,attrs**)**	
Plan	... prov:type='prov:Plan' ...	
Person	... prov:type='prov:Person' ...	
Organization	... prov:type='prov:Organization' ...	
SoftwareAgent	... prov:type='prov:SoftwareAgent' ...	
Influence	wasInfluencedBy(id;e2,e1,attrs)	
Bundle constructor	bundle id description_1 ... description_n endBundle	Component 4: Bundles
Bundle type	... prov:type='prov:Bundle' ...	
Alternate	alternateOf(alt1, alt2)	Component 5: Alternate
Specialization	specializationOf(infra, supra)	
Collection	... prov:type='prov:Collection' ...	Component 6: Collections
EmptyCollection	... prov:type='prov:EmptyCollection' ...	
Membership	hadMember(c,e)	

https://www.w3.org/TR/2013/REC-prov-dm-20130430/#prov-dm-types-and-relations-fig

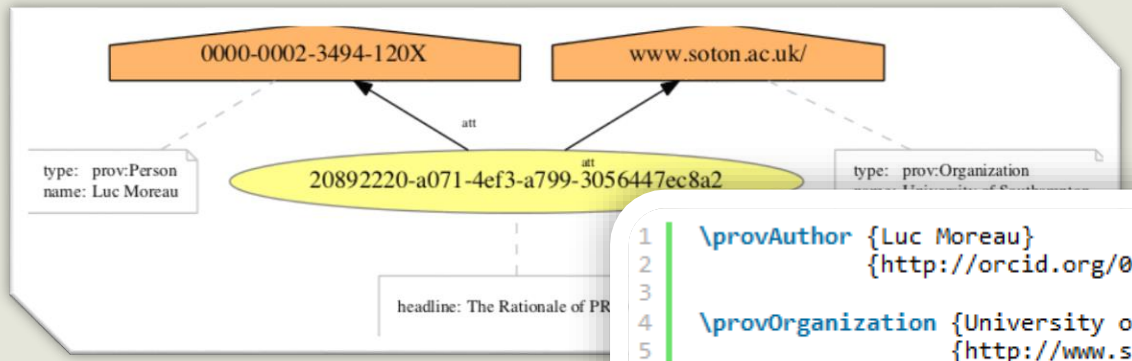
W3C PROV. Ejemplos

https://data.globalchange.gov/image/1a061197-95cf-47bd-9db4-f661c711a174

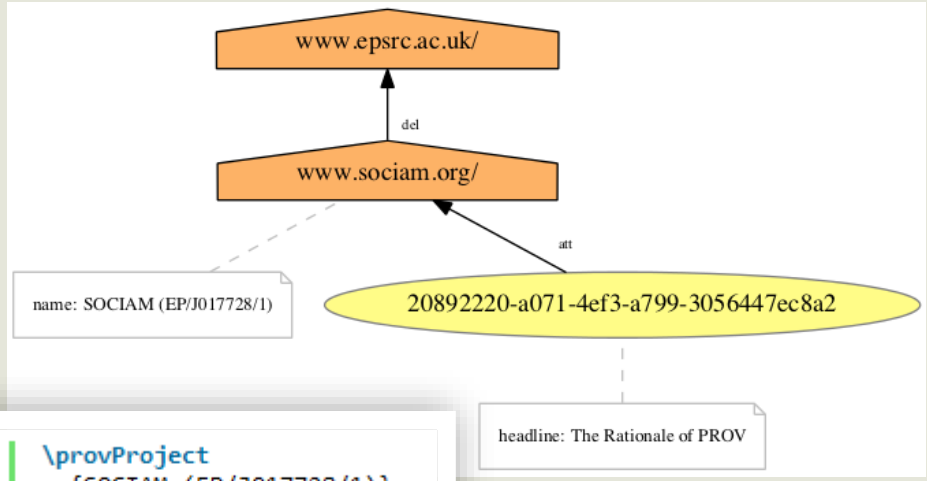


W3C PROV. Ejemplos

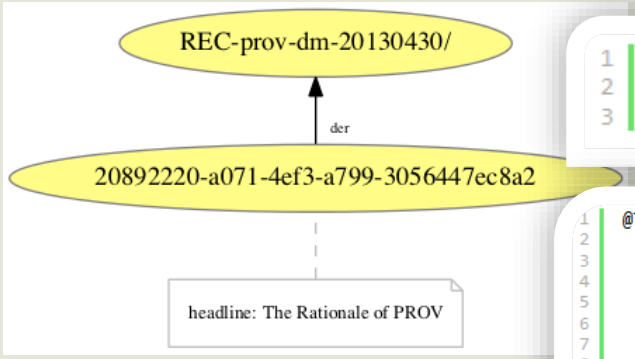
https://lucmoreau.wordpress.com/2015/06/24/provenance-of-publications-a-prov-style-for-latex/




```
\provAuthor {Luc Moreau}
            {http://orcid.org/0000-0002-3494-120X}

\provOrganization {University of Southampton}
                  {http://www.soton.ac.uk/}

\provTitle {The Rationale of PROV}
```



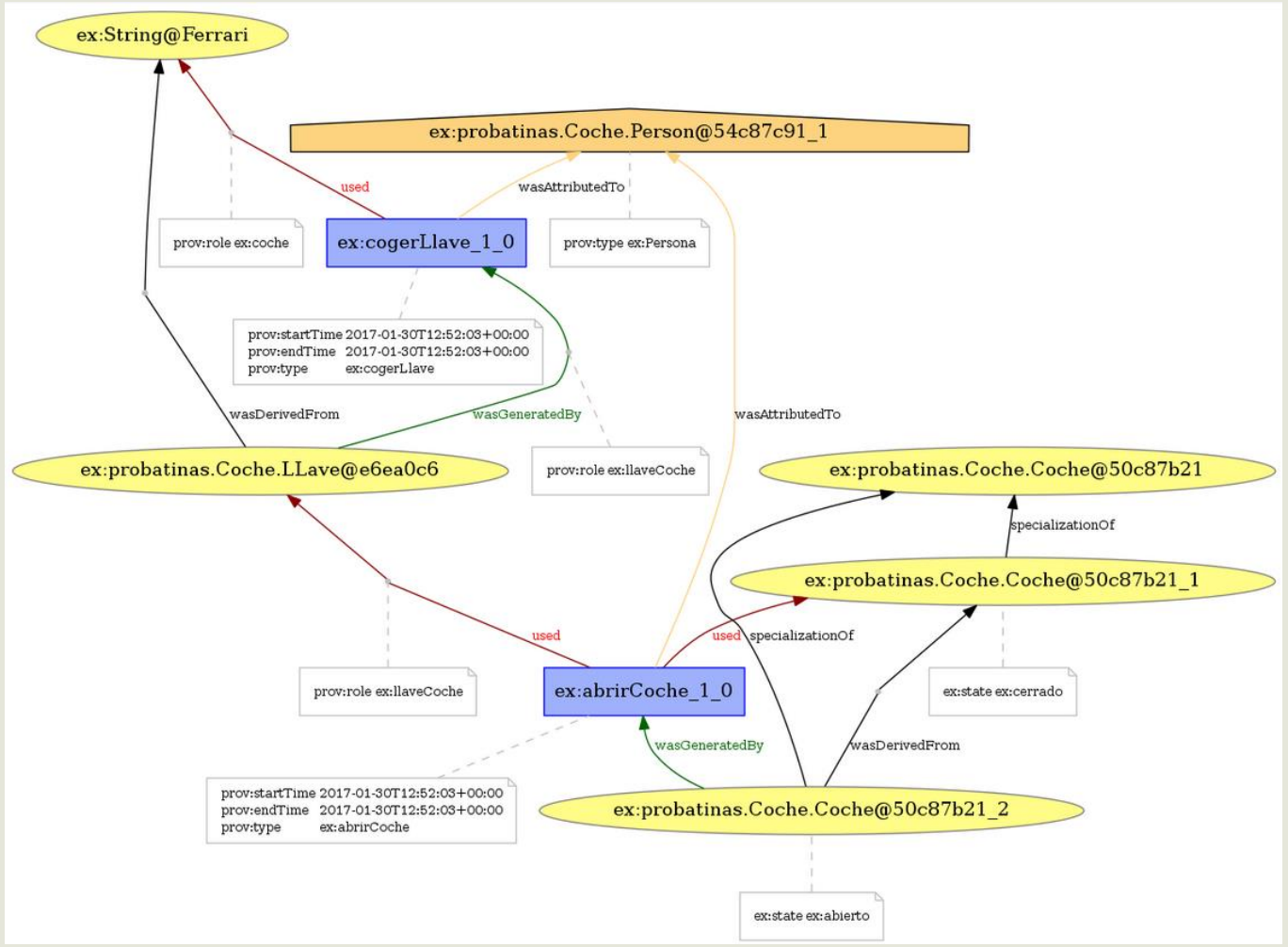

```
\provProject
  {SOCIAM (EP/J017728/1)}
  {http://www.sociam.org/}
  {http://www.epsrc.ac.uk/}
```




```
\provBibliography
\bibliographystyle{elsarticle}
\bibliography{rationale}
```

```
@TechReport{prov-dm:20130430,
  author = {Luc Moreau and Paolo {Missier (eds.)} …},
  title = {PROV-DM: The PROV Data Model},
  institution = {World Wide Web Consortium},
  year = {2013},
  type = {W3C Recommendation},
  number = {REC-prov-dm-20130430},
  month = oct,
  url = {http://www.w3.org/TR/2013/REC-prov-dm-20130430/}}
```

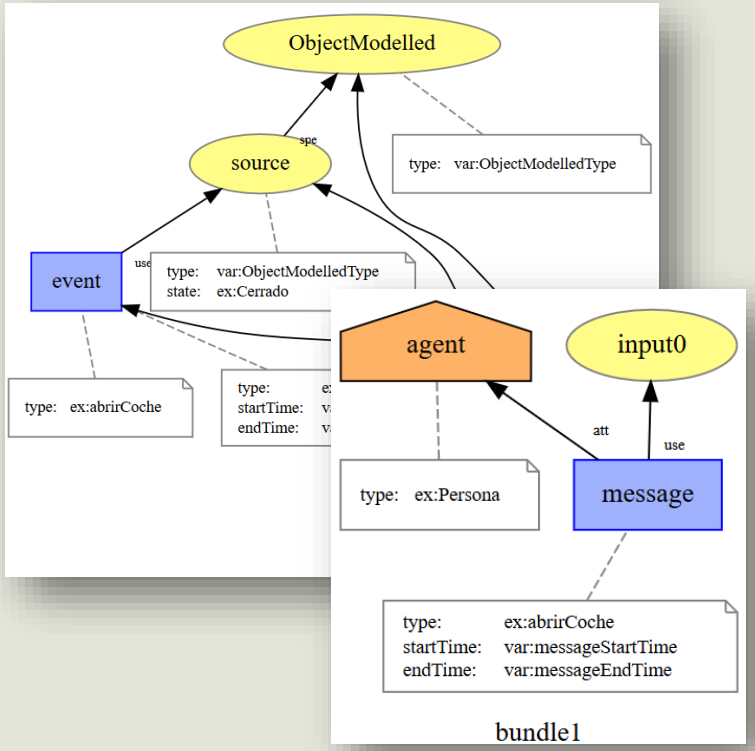
W3C PROV Document



Índice

- Definición de provenance
- Ejemplos de provenance
- Objetivo
- W3C PROV standard
- **PROV-Templates**
- De UML a PROV
  - UML Sequence Diagrams
  - UML State Diagrams
- Extracción de Bindings
- Caso de estudio

PROV-Templates

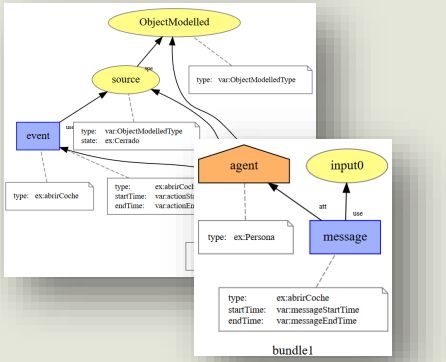


Templates

```
document
prefix tmpl <http://openprovenance.org/tmpl#>
prefix var <http://openprovenance.org/var#>
prefix ex <http://example.org/>


entity(var:ObjectModelled,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21', tmpl:2dvalue_0_0 = 'ex:Coche'])

entity(var:input0,[tmpl:value_0 = 'ex:probatinas.Coche.LLave@e6ea0c6'])

entity(var:message,[tmpl:value_0 = 'ex:abrirCoche_1_0'])
entity(var:messageStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:messageEndTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

entity(var:event,[tmpl:value_0 = 'ex:abrirCoche_1_0'])
entity(var:eventStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:eventEndTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

entity(var:agent,[tmpl:value_0 = 'ex:probatinas.Coche.Person@54c87c91_1'])

entity(var:source,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21_1'])
entity(var:target,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21_2'])

endDocument
```

Bindings

PROV-Template



Usa

Expansión

Usa

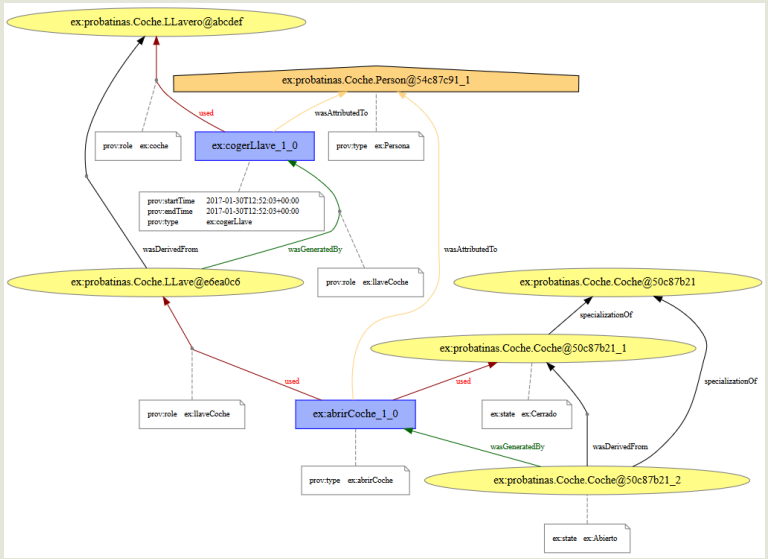
```
document
prefix tmpl <http://openprovenance.org/tmpl#>
prefix var <http://openprovenance.org/var#>
prefix ex <http://example.org/>

entity(var:ObjectModelled,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21', tmpl:2dvalue_0_0 = 'ex:Coche'])

entity(var:input0,[tmpl:value_0 = 'ex:probatinas.Coche.LLave@e6ea0c6'])

entity(var:message,[tmpl:value_0 = 'ex:abrirCoche_1_0'])
entity(var:messageStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:messageEndTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

entity(var:event,[tmpl:value_0 = 'ex:abrirCoche_1_0'])
entity(var:eventStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:eventEndTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

entity(var:agent,[tmpl:value_0 = 'ex:probatinas.Coche.Person@54c87c91_1'])

entity(var:source,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21_1'])
entity(var:target,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21_2'])

endDocument
```

genera

Templates

Bindings

PROV document



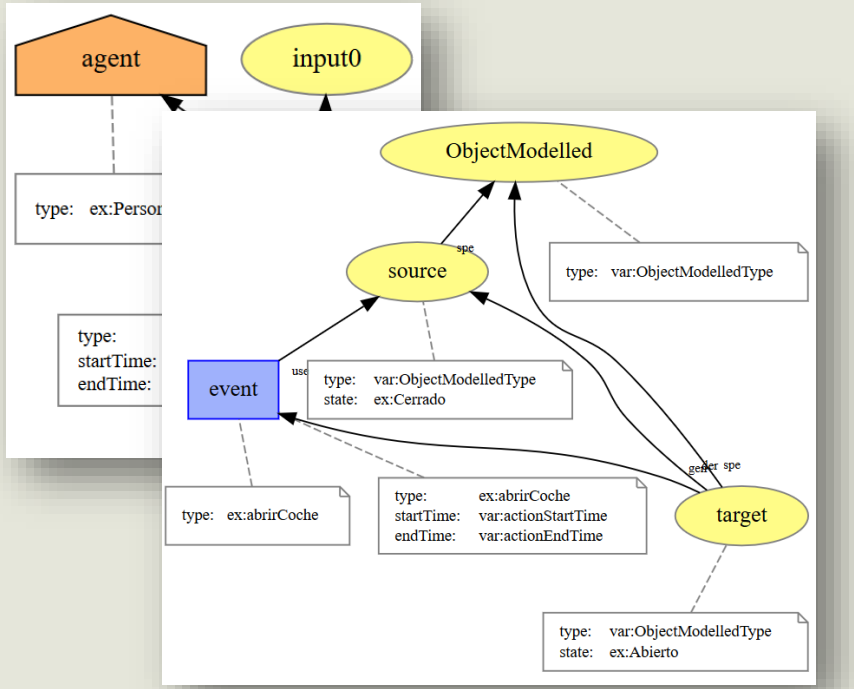
PROV-Template
Bindings

Bindings

```
document
prefix tmpl <http://openprovenance.org/tmpl#>
prefix var <http://openprovenance.org/var#>
prefix ex <http://example.org/>


entity(var:ObjectModelled,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21', tmpl:2dvalue_0_0 = 'ex:Coche'])

entity(var:input0,[tmpl:value_0 = 'ex:probatinas.Coche.LLave@e6ea0c6'])

entity(var:message,[tmpl:value_0 = 'ex:abrirCoche_1_0'])
entity(var:messageStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:messageEndTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

entity(var:event,[tmpl:value_0 = 'ex:abrirCoche_1_0'])
entity(var:eventStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:eventEndTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

entity(var:agent,[tmpl:value_0 = 'ex:probatinas.Coche.Person@54c87c91_1'])

entity(var:source,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21_1'])
entity(var:target,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21_2'])

endDocument
```

PROV-Template
Templates

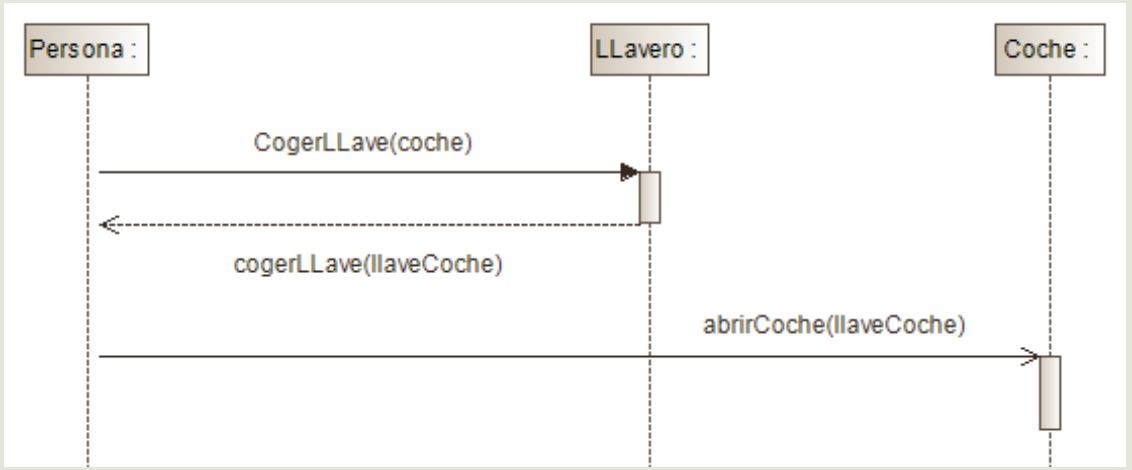
Templates

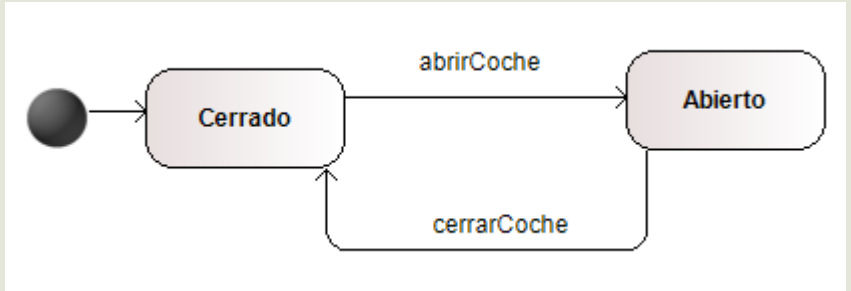


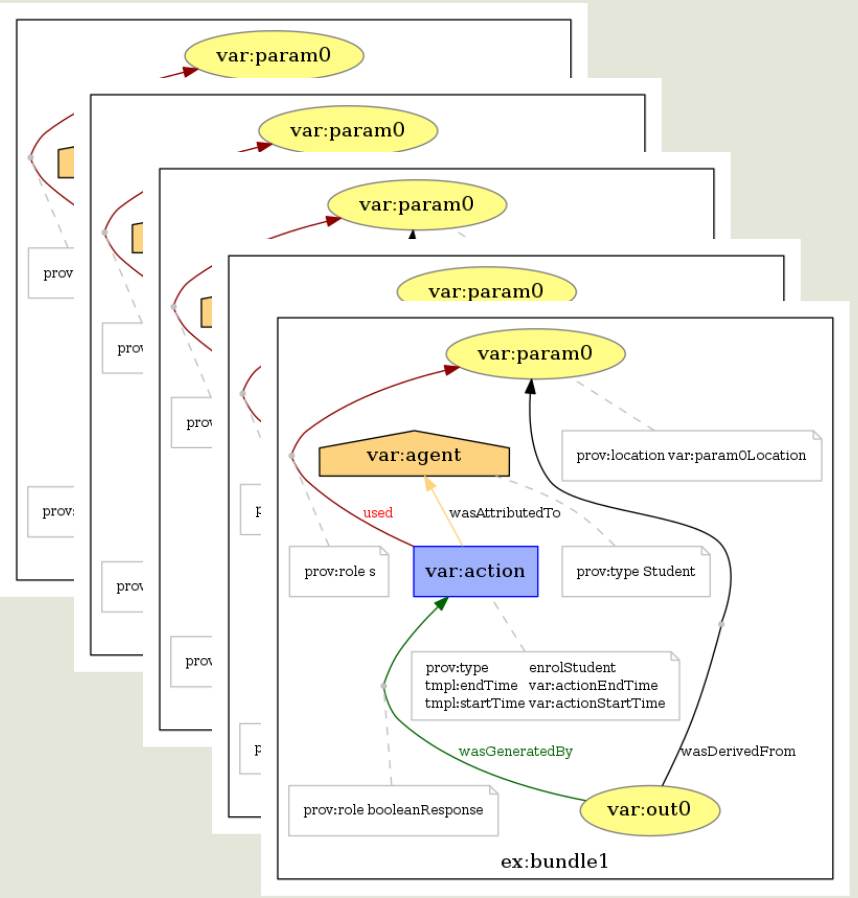

```
document

prefix prov <http://www.w3.org/ns/prov#>
prefix tmpl <http://openprovenance.org/tmpl#>
prefix var <http://openprovenance.org/var#>
prefix ex <http://example.org/>

    bundle ex:bundle1
        entity(var:ObjectModelled, [prov:type='var:ObjectModelledType'])
        entity(var:target, [prov:type='var:ObjectModelledType' , ex:state= 'ex:Abierto'])
        specializationOf(var:target,var:ObjectModelled)
        wasGeneratedBy(var:target, var:event, -)
        wasDerivedFrom(var:target,var:source)

         entity(var:source, [prov:type='var:ObjectModelledType' , ex:state= 'ex:Cerrado'])
         activity(var:event, [prov:type = 'ex:abrirCoche',
           tmpl:startTime = 'var:actionStartTime', tmpl:endTime = 'var:actionEndTime' ])

         activity(var:event, [prov:type = 'ex:abrirCoche'])

         used(var:event, var:source, -)
         specializationOf(var:source,var:ObjectModelled)
    endBundle
endDocument
```

Índice

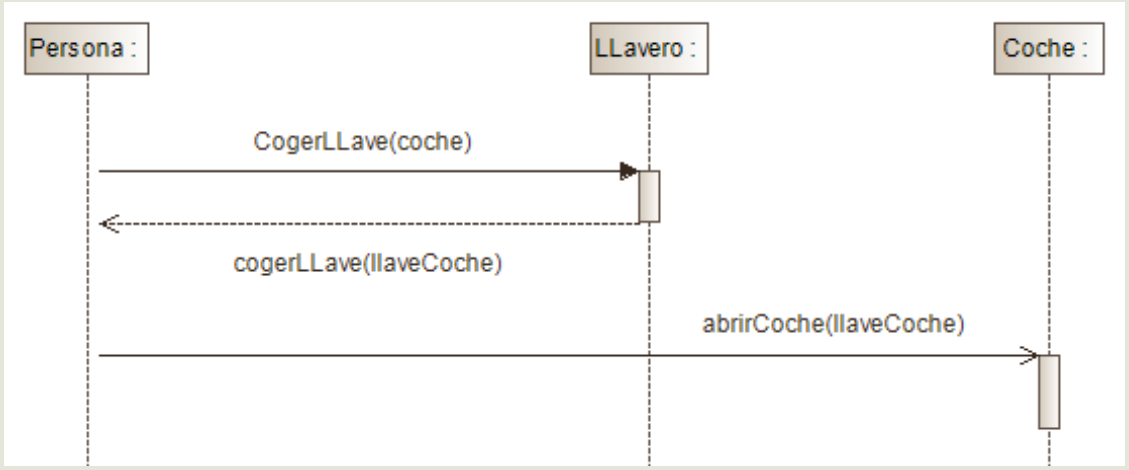
UML Sequence Diagrams
Objetivo

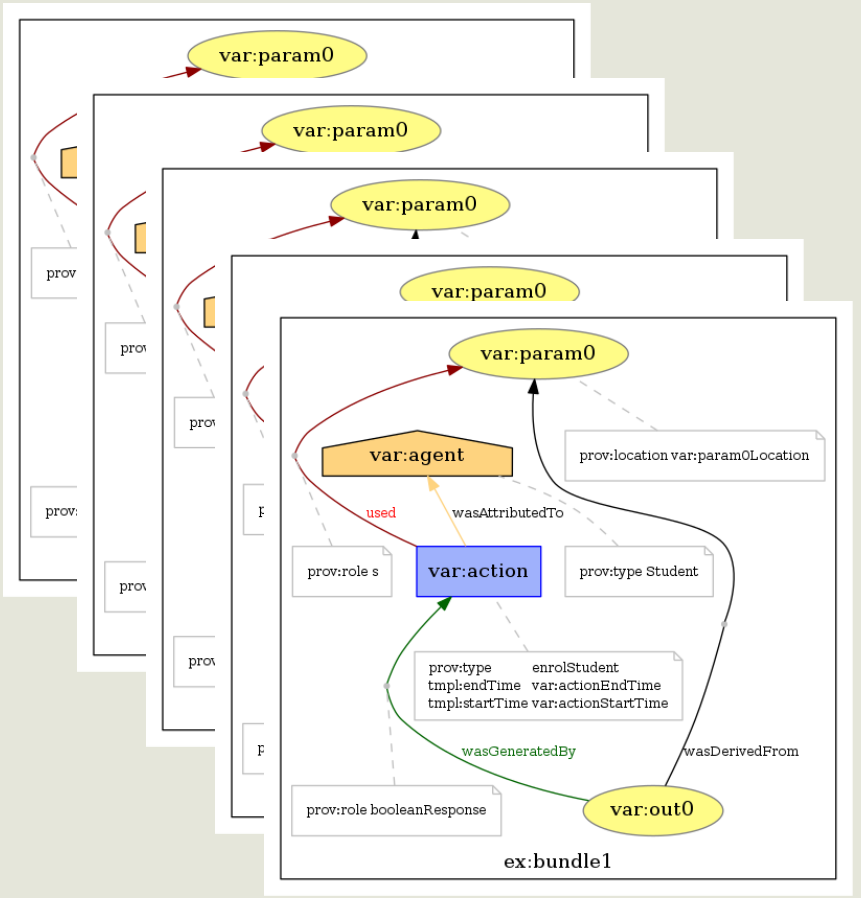




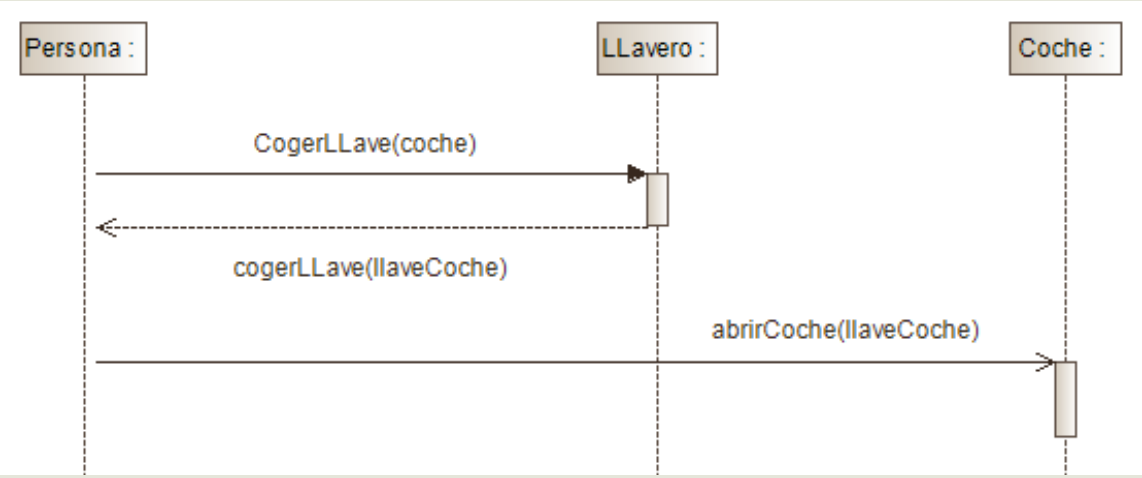


UML Sequence Diagrams
Objetivo

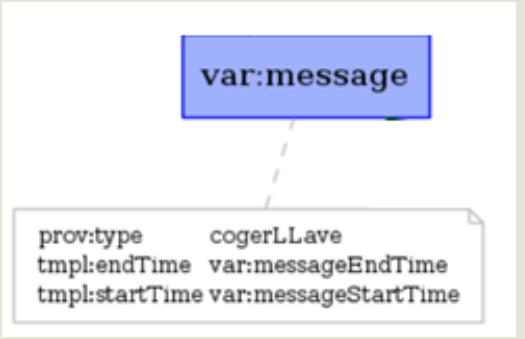




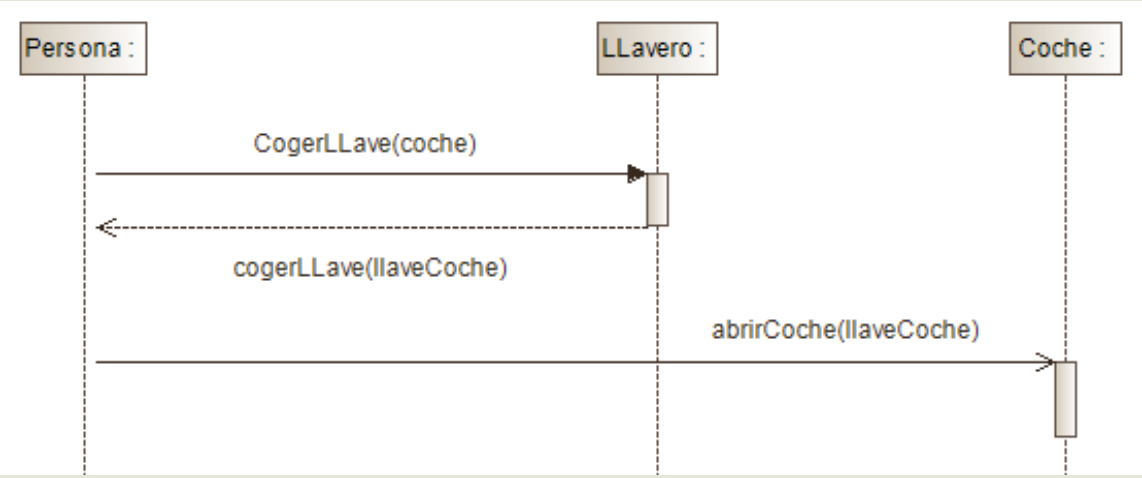
UML Sequence Diagrams
Reglas de transformación. SeqR1



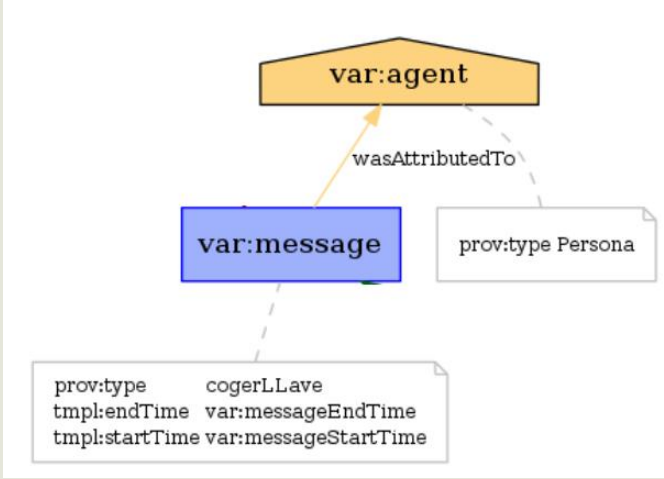
SeqR1. Cada mensaje (no reply) se traduce a una Activity



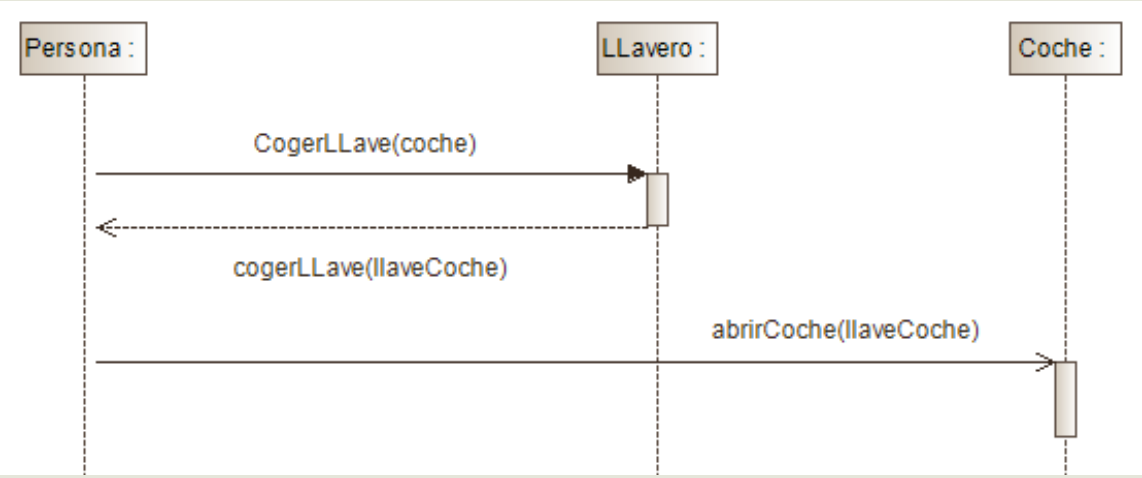
UML Sequence Diagrams
Reglas de transformación. SeqR2



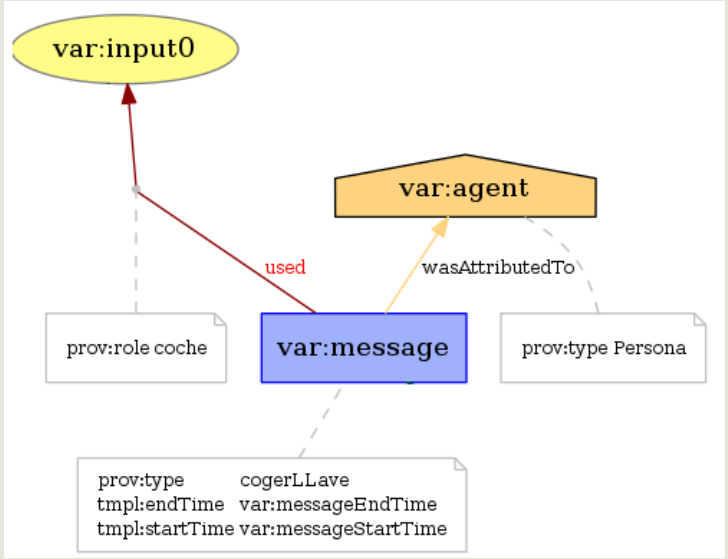
SeqR2. Cada LifeLine que representa el envío de un Mensaje es traducido como un Agente



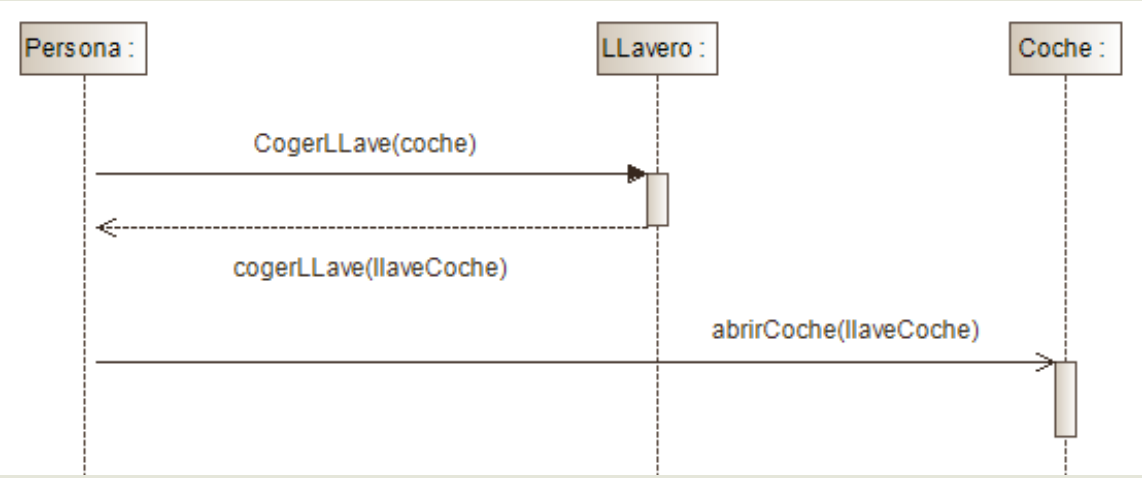
UML Sequence Diagrams
Reglas de transformación. SeqR3



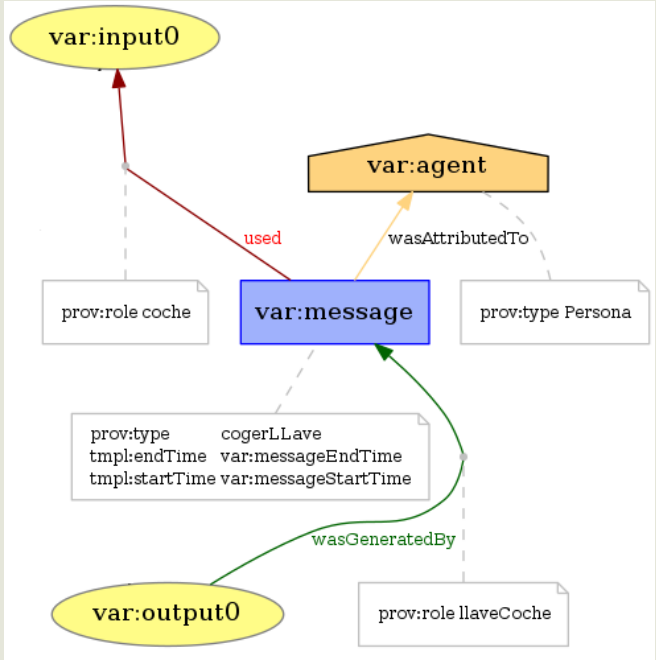
SeqR3. Cada argumento input en un mensaje asíncrono o síncrono es traducido como una entidad



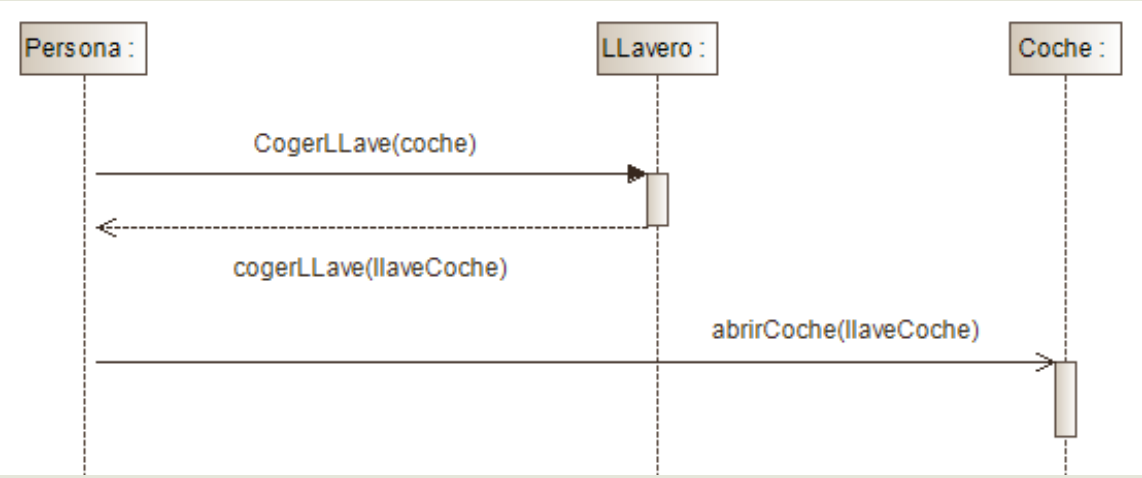
UML Sequence Diagrams
Reglas de transformación. SeqR4



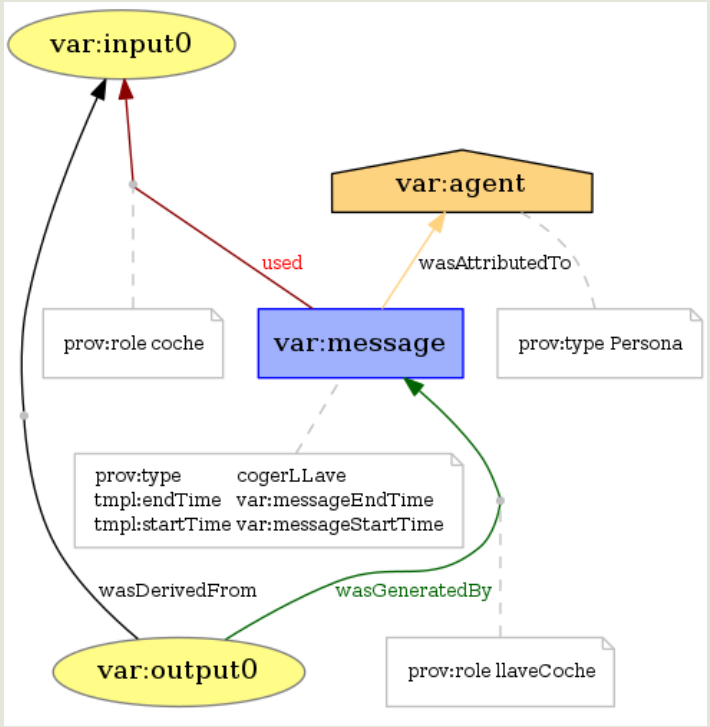
SeqR4. Cada argumento output en un mensaje reply traducido como una entidad



UML Sequence Diagrams
Reglas de transformación. SeqR5



SeqR5. Creación de la relación wasDerivedFrom



UML Sequence Diagrams. Algoritmo

SeqR1. Cada mensaje se traduce a una Activity

SeqR2. Cada LifeLine que representa el envío de un mensaje es traducido como un Agente

SeqR3. Cada argumento input en un mensaje asíncrono o síncrono es traducido como una entidad

SeqR4. Cada argumento output en un mensaje reply traducido como una entidad

SeqR5. Creación de la relación wasDerivedFrom

```
1 procedure  Seq2PROV
2 forEach message in SqD do
3   print  activity(var:message,[prov:type ="<messageName>",
4       tmpl:startTime= 'var:messageStartTime',
5       tmpl:endTime = 'var:messageEndTime' ])
6   print  agent(var:agent, [prov:type = "<LifeLineName>"])
7   print  wasAttributedTo(var:message, var:agent)
8   List inParamsList = message.arguments
9   foreach  inputN  in inParamsList do
10   print  entity(var:inputN, [prov:type ="nameInputN"])
11   print  used(var:message, var:inputN,-,
12                              [prov:role="<paramName>"])
13  endForEach
14  if hasReply(message)
15    List outParamList=getReplyMsg(message).arguments
16    foreach outputN in outParamsList do
17      print  entity(var:outputN,[prov:type="nameOutputN"])
18      print  wasGeneratedBy(var:outputN, var:message,-,
19                              [prov:role="<paramName>"])
20      foreach  inputN in inParamsList do
21        print  wasDerivedFrom(var:outputN, var:inputN)
22      endForEach
23    endForEach
24  endIf
25 endForEach
26 endProcedure
```

UML Sequence Diagrams. Algoritmo

```
1 procedure  Seq2PROV
2 forEach message in SqD do
3   print  activity(var:message,[prov:type ="<messageName>"
4      tmpl:startTime= 'var:messageStartTime',
5      tmpl:endTime = 'var:messageEndTime' ])
6   print  agent(var:agent, [prov:type = "<LifeLineName>"])
7   print  wasAttributedTo(var:message, var:agent)
8   List inParamsList = message.arguments
9   foreach  inputN  in inParamsList do
10   print  entity(var:inputN, [prov:type ="nameInputN"])
11   print  used(var:message, var:inputN,-,
12                          [prov:role="<paramName>"])
13  endForEach
14  if hasReply(message)
15    List outParamList=getReplyMsg(message).arguments
16    foreach outputN in outParamsList do
17      print  entity(var:outputN,[prov:type="nameOutputN"])
18      print  wasGeneratedBy(var:outputN, var:message,-,
19                              [prov:role="<paramName>"])
20      foreach  inputN in inParamsList do
21        print  wasDerivedFrom(var:outputN, var:inputN)
22      endForEach
23    endForEach
24  endIf
25 endForEach
26 endProcedure
```

```
3   print  activity(var:message,[prov:type ="<messageName>",
4      tmpl:startTime= 'var:messageStartTime',
5      tmpl:endTime = 'var:messageEndTime' ])
6   print  agent(var:agent, [prov:type = "<LifeLineName>"])
7   print  wasAttributedTo(var:message, var:agent)
```

UML Sequence Diagrams. Algoritmo

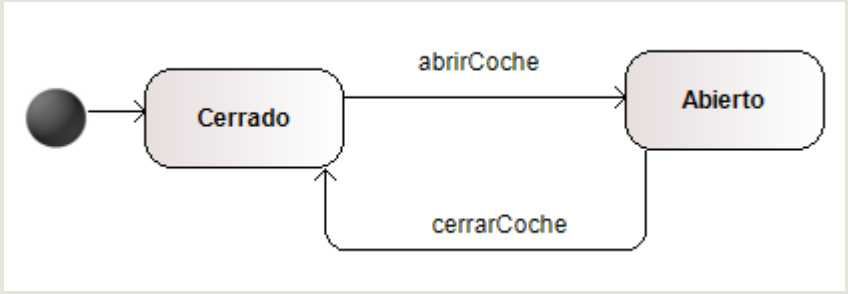
```
1 procedure  Seq2PROV
2 forEach message in SqD do
3   print  activity(var:message,[prov:type ="<messageName>"
4      tmpl:startTime= 'var:messageStartTime',
5      tmpl:endTime = 'var:messageEndTime' ])
6   print  agent(var:agent, [prov:type = "<LifeLineName>"])
7   print  wasAttributedTo(var:message, var:agent)
8   List inParamsList = message.arguments
9   foreach  inputN  in inParamsList do
10   print  entity(var:inputN, [prov:type ="nameInputN"])
11   print  used(var:message, var: inputN,-,
12                          [prov:role="<paramName>"])
13  endForEach
14  if hasReply(message)
15    List outParamList=getReplyMsg(message).arguments
16    foreach outputN in outParamsList do
17      print  entity(var:outputN,[prov:type="nameOutputN"])
18      print  wasGeneratedBy(var:outputN, var:message,-,
19                          [prov:role="<paramName>"])
20      foreach  inputN in inParamsList do
21        print  wasDerivedFrom(var:outputN, var:inputN)
22      endForEach
23    endForEach
24  endIf
25 endForEach
26 endProcedure
```

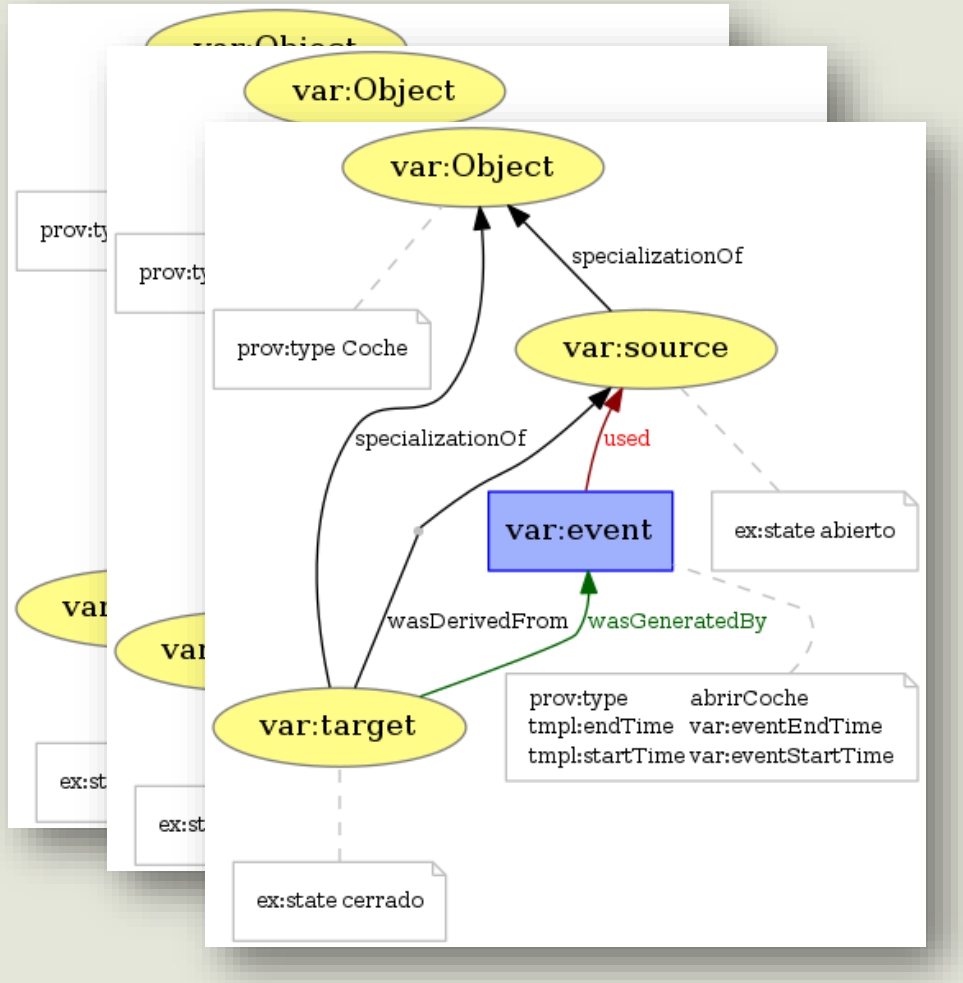
```
9   foreach  inputN  in inParamsList do
10   print  entity(var:inputN, [prov:type ="nameInputN"])
11   print  used(var:message, var: inputN,-,
12                          [prov:role="<paramName>"])
13  endForEach
```

```
16    foreach outputN in outParamsList do
17      print  entity(var:outputN,[prov:type="nameOutputN"])
18      print  wasGeneratedBy(var:outputN, var:message,-,
19                          [prov:role="<paramName>"])
20      foreach  inputN in inParamsList do
21        print  wasDerivedFrom(var:outputN, var:inputN)
22      endForEach
23    endForEach
```

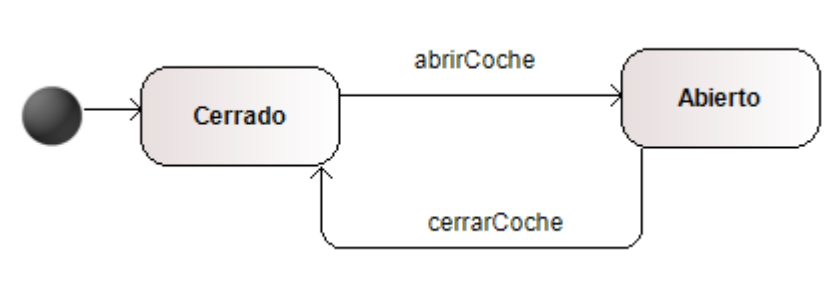
Índice

UML State Diagrams
Objetivo

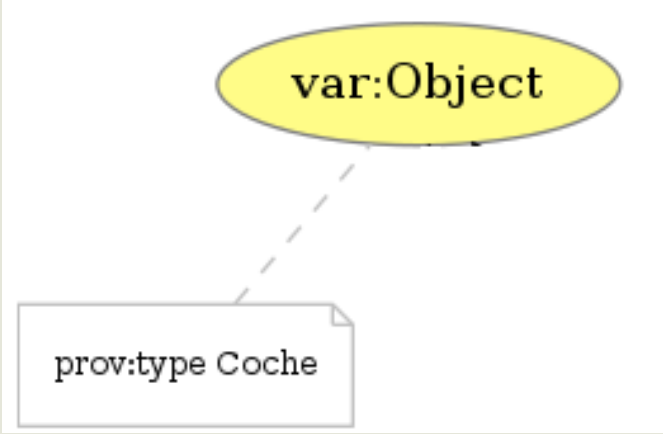




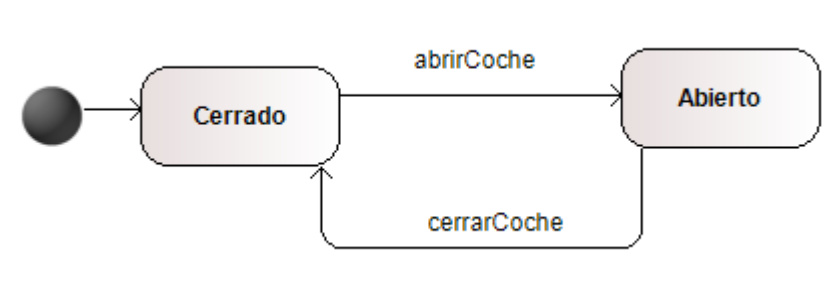
UML State Diagrams



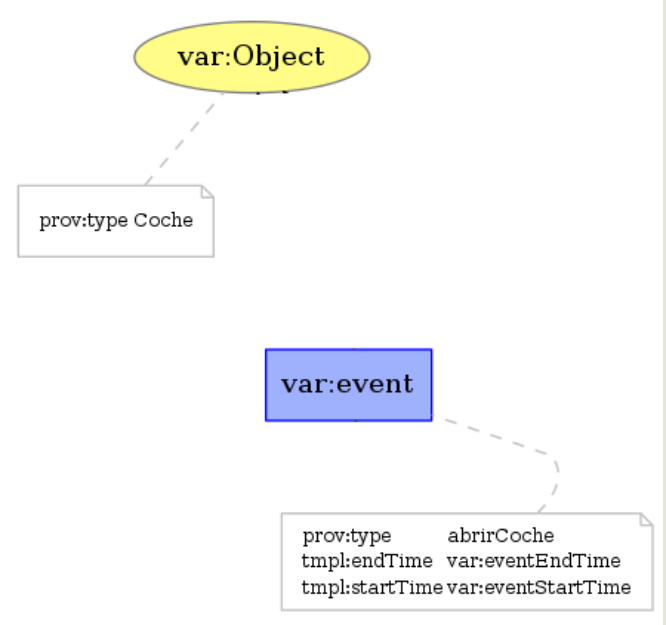
StR1. Objeto modelado con el diagrama de estados es considerado una entidad



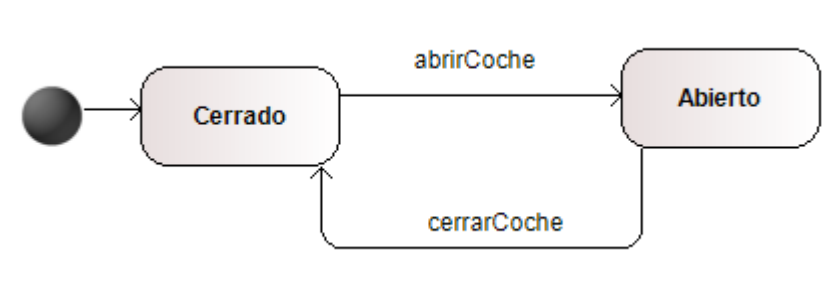
UML State Diagrams



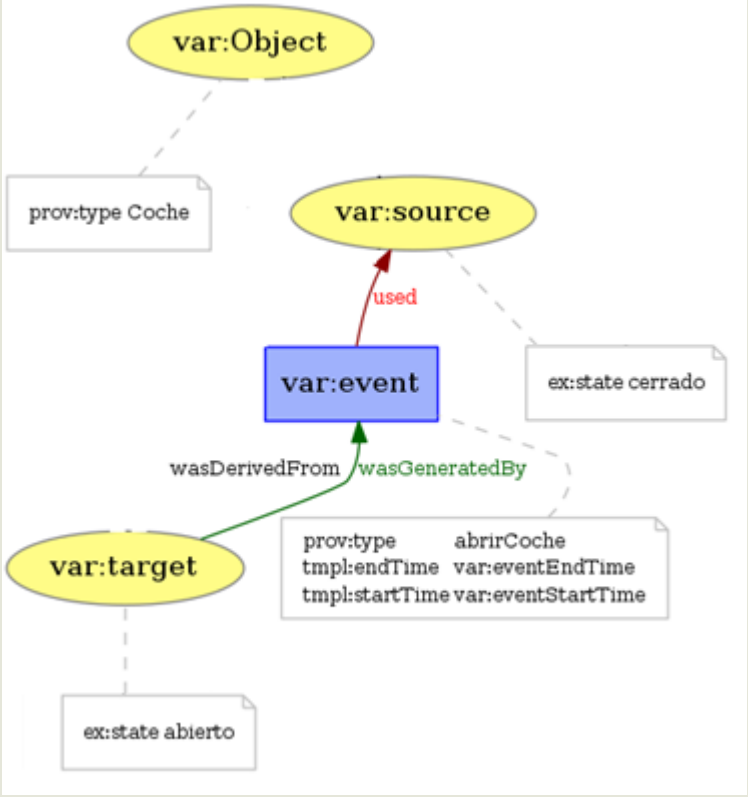
StR2. El evento dentro de una transición es traducido en una actividad



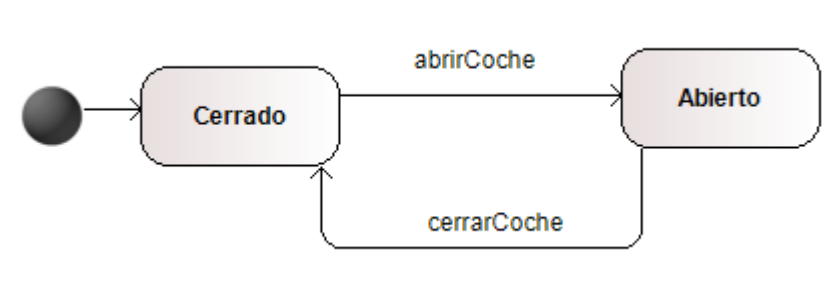
UML State Diagrams



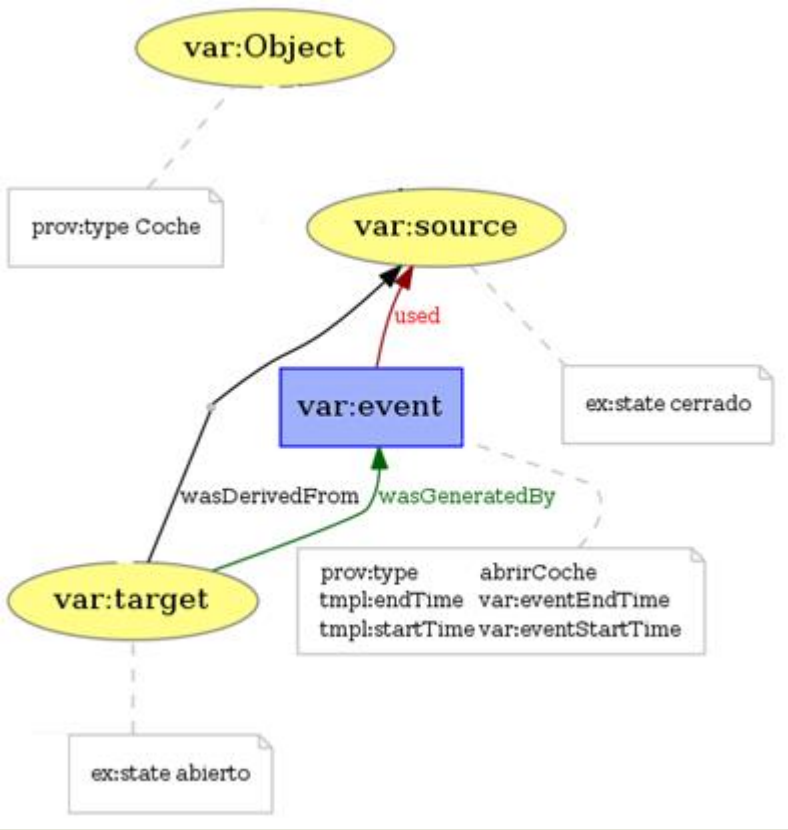
StR3. Cada estado es traducido en una entidad



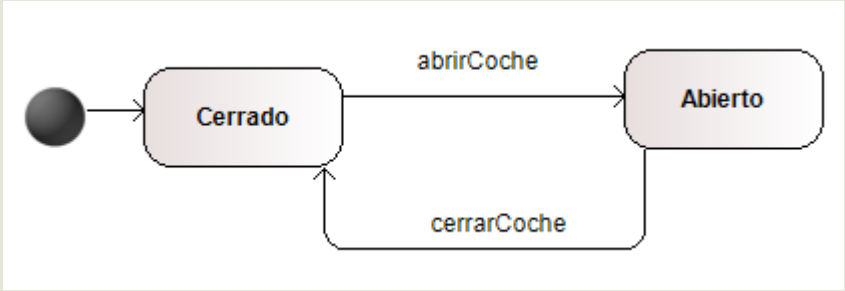
UML State Diagrams



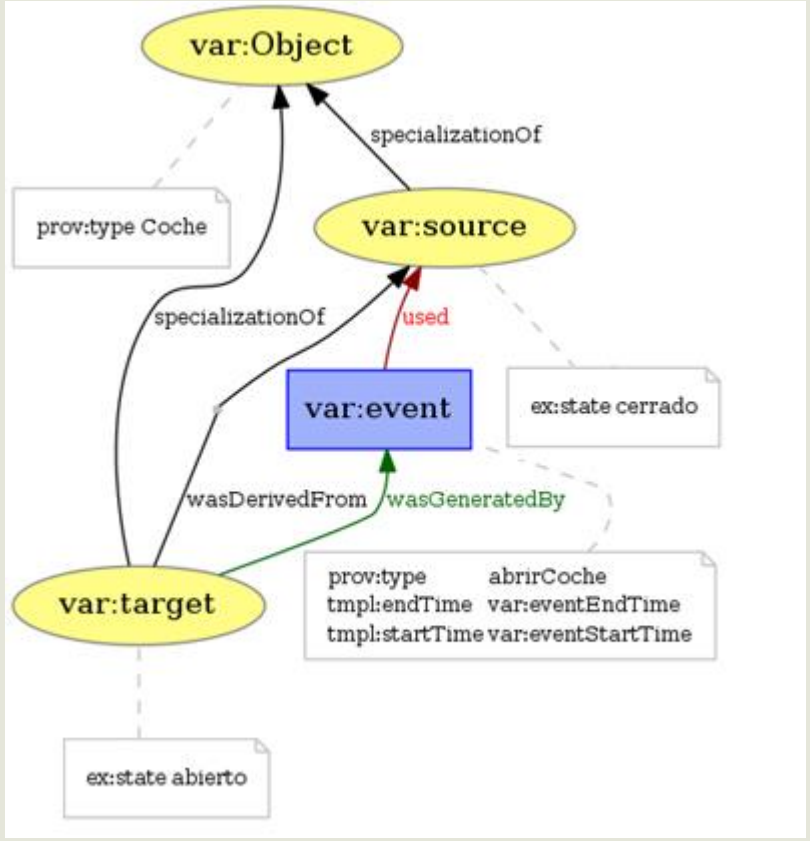
StR4. Creación de la relación wasDerivedFrom



UML State Diagrams



StR5. ¿Dentro de un estado Compuesto?



UML State Diagrams. Algoritmo

StR1. Objeto modelado con el diagrama de estados es Considerado una entidad

StR2. El evento dentro de una transición es traducido en una actividad

StR3. Cada estado es traducido en una entidad

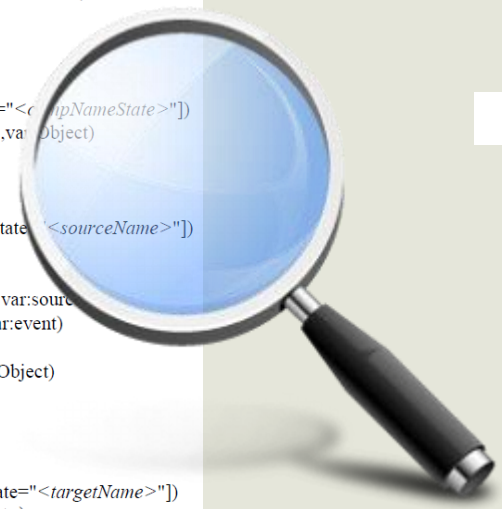
StR4. Creación de la relación wasDerivedFrom

StR5. ¿Dentro de un estado Compuesto?

```
7   if transition is within a state
8     print entity(var:<compName>,
9           [prov:type="<objectName>",u2p:state="<compNameState>"
10    print specializationOf(var:<compName>,var:Object)
11  endIf
12  if typeof transition.source= ="State"
13    print entity(var:source,
14          [prov:type="<objectName>",u2p:state="<sourceName>"
15    print used(var:event,var:source,-)
16    if transition is within a state
17      print hadMember(var:<compName>, var:source)
18      print wasInvalidatedBy(var:source, var:event)
19    else
20      print specializationOf(var:source,var:Object)
21    endIf
22  endIf
23  if typeof transition.target= ="State"
24    print entity(var:target,
25          [prov:type="<objectName>",u2p:state="<targetName>"])
26    print wasGeneratedBy(var:event,var:target,-)
27    if transition.source = = "State"
28      print wasDerivedFrom(var:target,var:source)
29    endIf
30    if transition is within a state
```

UML State Diagrams. Algoritmo

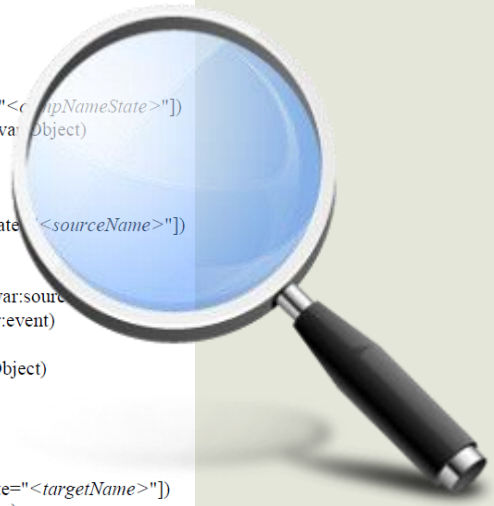
```
1 procedure SMD2PROV
2 forEach  message in  SMD do
3   print  entity(var:Object, [prov:type="<objectName>"])
4   print  activity(var:event, [prov:type ="<EventName>",
5    tmpl:startTime= 'var:eventStartTime',
6    tmpl:endTime = 'var:eventEndTime' ])
7   if transition is within a state
8    print  entity(var:<compName>,
9          [prov:type="<objectName>",u2p:state="<compNameState>"])
10    print  specializationOf(var:<compName>,var:Object)
11  endIf
12  if  typeof transition.source= ="State"
13    print  entity(var:source,
14           [prov:type="<objectName>",u2p:state="<sourceName>"])
15    print  used(var:event,var:source,-)
16    if  transition is within a state
17      print  hadMember(var:<compName>, var:source)
18      print  wasInvalidatedBy(var:source, var:event)
19    else
20      print  specializationOf(var:source,var:Object)
21    endIf
22  endIf
23  if  typeof transition.target= ="State"
24   print  entity(var:target,
25          [prov:type="<objectName>",u2p:state="<targetName>"])
26   print  wasGeneratedBy(var:event,var:target,-)
27   if transition.source = = "State"
28     print  wasDerivedFrom(var:target,var:source)
29   endIf
30   if transition is within a state
31     print  hadMember(var:<compName>, var:target)
32   else
33     print  specializationOf(var:target,var:Object)
34   endIf
35  endIf
36 endForEach
37 endProcedure
```



3 **print** entity(var:Object, [prov:type="<*objectName*>"])

UML State Diagrams. Algoritmo

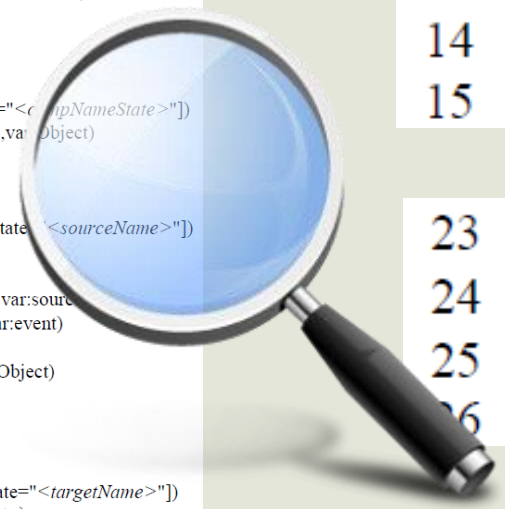
```
1 procedure SMD2PROV
2 forEach message in SMD do
3   print entity(var:Object, [prov:type="<objectName>"])
4   print activity(var:event, [prov:type ="<EventName>",
5    tmpl:startTime= 'var:eventStartTime',
6    tmpl:endTime = 'var:eventEndTime' ])
7   if transition is within a state
8    print entity(var:<compName>,
9         [prov:type="<objectName>",u2p:state="<
10    print specializationOf(var:<compName>,va
11  endIf
12  if typeof transition.source= ="State"
13    print entity(var:source,
14          [prov:type="<objectName>",u2p:state
15    print used(var:event,var:source,-)
16    if transition is within a state
17      print hadMember(var:<compName>, var:sour
18      print wasInvalidatedBy(var:source, var:event)
19    else
20      print specializationOf(var:source,var:Object)
21    endIf
22  endIf
23  if typeof transition.target= ="State"
24   print entity(var:target,
25         [prov:type="<objectName>",u2p:state="<targetName>"])
26   print wasGeneratedBy(var:event,var:target,-)
27   if transition.source = = "State"
28     print wasDerivedFrom(var:target,var:source)
29   endIf
30   if transition is within a state
31     print hadMember(var:<compName>, var:target)
32   else
33     print specializationOf(var:target,var:Object)
34   endIf
35  endIf
36 endForEach
37 endProcedure
```



```
4   print activity(var:event, [prov:type ="<EventName>",
5    tmpl:startTime= 'var:eventStartTime',
6    tmpl:endTime = 'var:eventEndTime' ])
```

UML State Diagrams. Algoritmo

```
1 procedure SMD2PROV
2 forEach message in SMD do
3   print entity(var:Object, [prov:type="<objectName>"])
4   print activity(var:event, [prov:type ="<EventName>",
5    tmpl:startTime= 'var:eventStartTime',
6    tmpl:endTime = 'var:eventEndTime' ])
7   if transition is within a state
8    print entity(var:<compName>,
9         [prov:type="<objectName>",u2p:state="<compNameState>"])
10    print specializationOf(var:<compName>,var:Object)
11  endIf
12  if typeof transition.source= ="State"
13    print entity(var:source,
14          [prov:type="<objectName>",u2p:state="<sourceName>"])
15    print used(var:event,var:source,-)
16    if transition is within a state
17      print hadMember(var:<compName>, var:source)
18      print wasInvalidatedBy(var:source, var:event)
19    else
20      print specializationOf(var:source,var:Object)
21    endIf
22  endIf
23  if typeof transition.target= ="State"
24   print entity(var:target,
25         [prov:type="<objectName>",u2p:state="<targetName>"])
26   print wasGeneratedBy(var:event,var:target,-)
27   if transition.source = = "State"
28     print wasDerivedFrom(var:target,var:source)
29   endIf
30   if transition is within a state
31     print hadMember(var:<compName>, var:target)
32   else
33     print specializationOf(var:target,var:Object)
34   endIf
35  endIf
36 endForEach
37 endProcedure
```



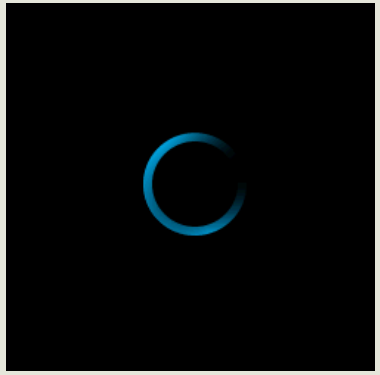
```
12  if typeof transition.source= ="State"
13    print entity(var:source,
14          [prov:type="<objectName>",u2p:state="<sourceName>"])
15    print used(var:event,var:source,-)
```

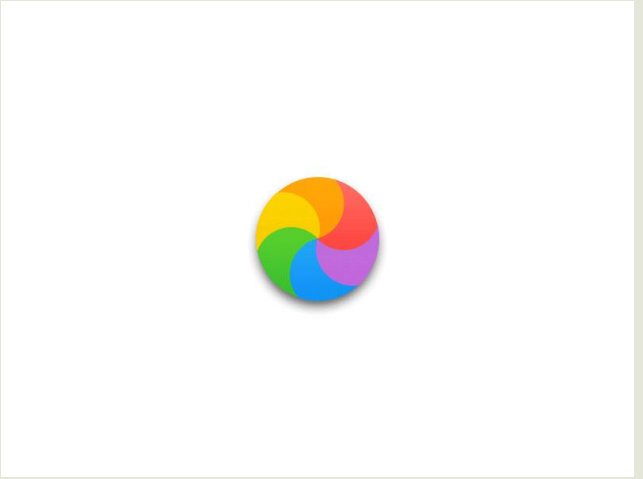
```
23  if typeof transition.target= ="State"
24   print entity(var:target,
25         [prov:type="<objectName>",u2p:state="<targetName>"])
26   print wasGeneratedBy(var:event,var:target,-)
```

Índice

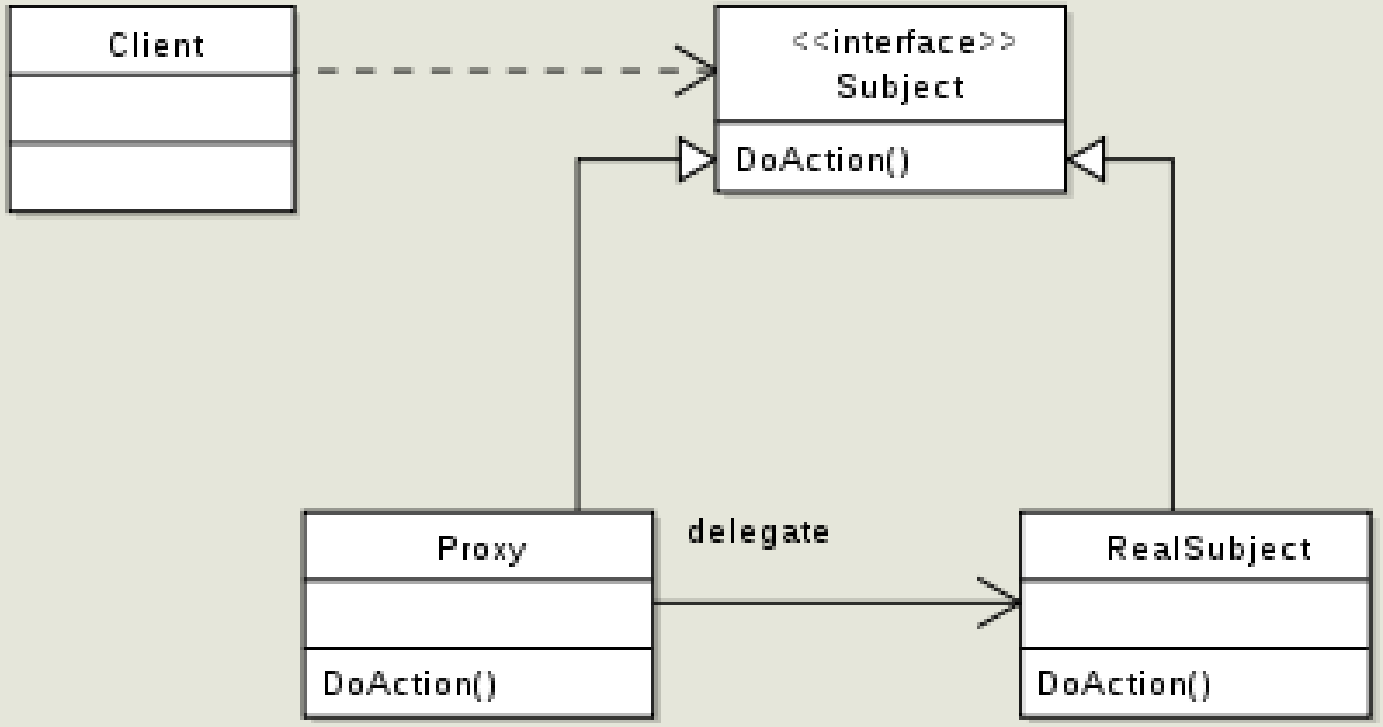
- Definición de provenance
- Ejemplos de provenance
- Objetivo
- W3C PROV standard
- PROV-Templates
- De UML a PROV
  - UML Sequence Diagrams
  - UML State Diagrams
- **Extracción de Bindings**
- Caso de estudio

¿Cómo capturar Bindings?





Proxy-Pattern



Dynamic Proxy

```
Coche ferrari = new Coche();
LLavero llavero = new LLavero();

LLave llaveCoche = llavero.cogerLlave( coche: "Ferrari");

ferrari.abrirCoche(llaveCoche);
```

```
ICoche ferrari = (ICoche) ProxyProv.generateProxy(new Coche());
ILLavero llavero = (ILLavero) ProxyProv.generateProxy(new LLavero());

LLave llaveCoche = llavero.cogerLlave( coche: "Ferrari");

ferrari.abrirCoche(llaveCoche);
```

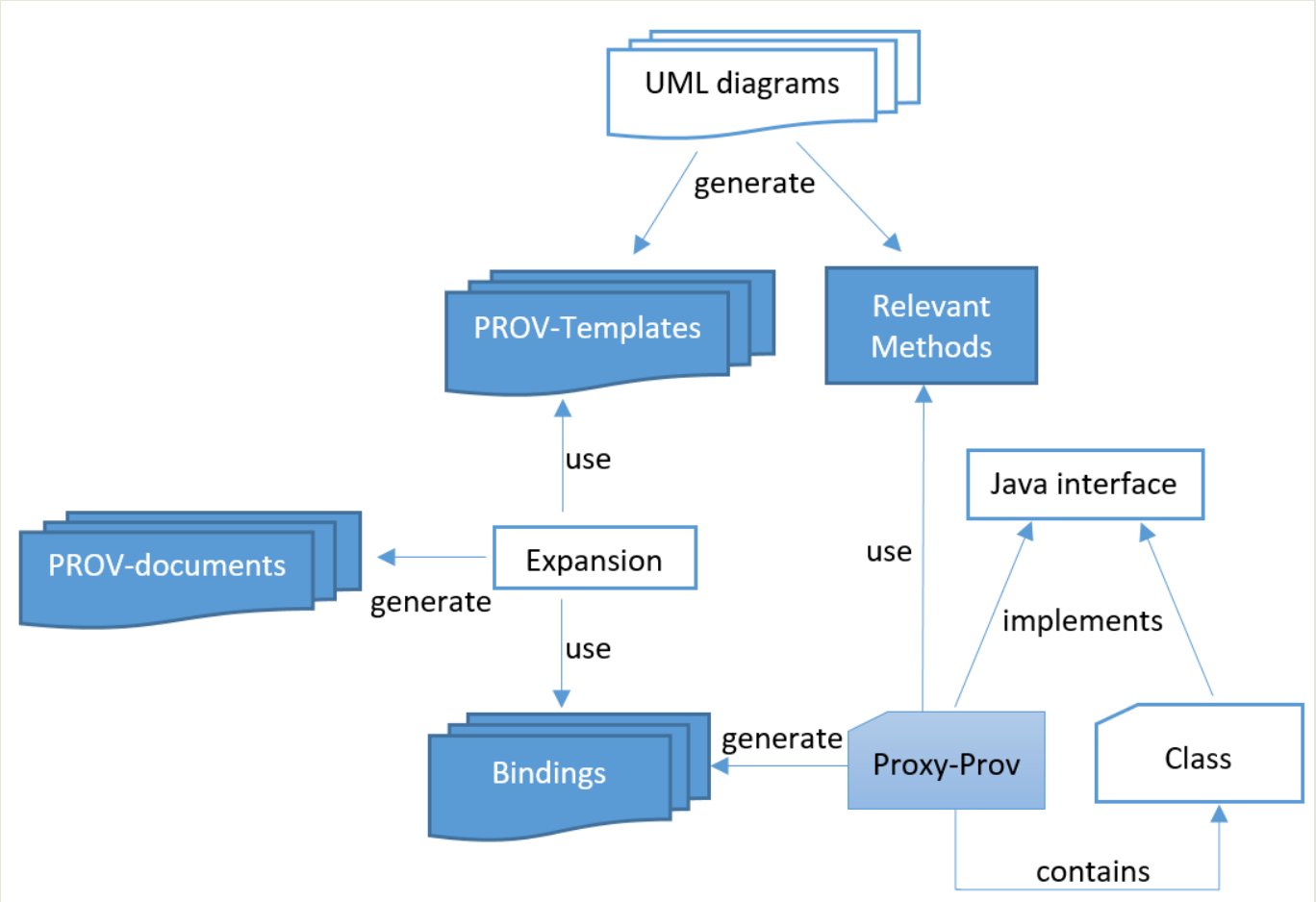
Dynamic Proxy

```
////////////////////////////////   activityEntity    ////////////////////////////////
Entity activityEntity = pfactory.newEntity(var( param: "message"));
activityEntity.getOther().add(pfactory.newOther(tmpl( param: "value_0"), ex( param: m.getName() + "_" + generateIdentifier(m.getName()) + "_" + getC
listEntities.add(activityEntity);
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//Invocation the method capturing start and end Date
SimpleDateFormat xsdDateTime = new SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:ss");
//start It is important 2dvalue_0_0 when the variable is an attribute
Entity startTimeEntity = pfactory.newEntity(var( param: "messageStartTime"));
startTimeEntity.getOther().add(pfactory.newOther(tmpl( param: "2dvalue_0_0"), xsdDateTime.format(start), pfactory.getName().XSD_DATETIME));
listEntities.add(startTimeEntity);

//end It is important 2dvalue_0_0 when the variable is an attribute
Entity endTimeEntity = pfactory.newEntity(var( param: "messageEndTime"));
endTimeEntity.getOther().add(pfactory.newOther(tmpl( param: "2dvalue_0_0"), xsdDateTime.format(end), pfactory.getName().XSD_DATETIME));
listEntities.add(endTimeEntity);
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

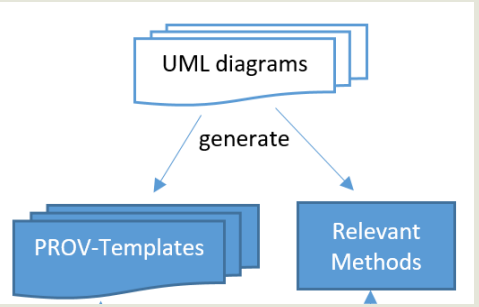
Índice

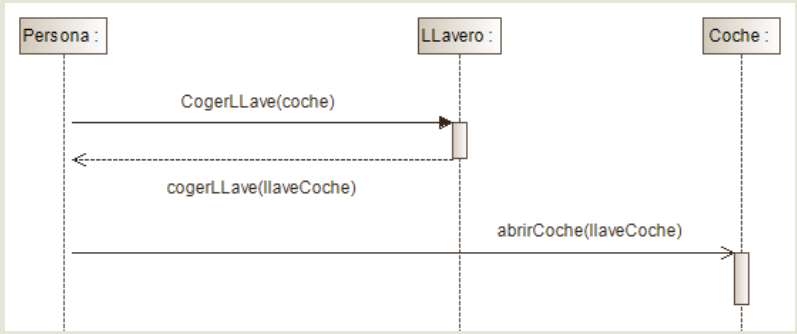
- Definición de provenance
- Ejemplos de provenance
- Objetivo
- W3C PROV standard
- PROV-Templates
- De UML a PROV
  - UML Sequence Diagrams
  - UML State Diagrams
- Extracción de Bindings
- **Caso de estudio**

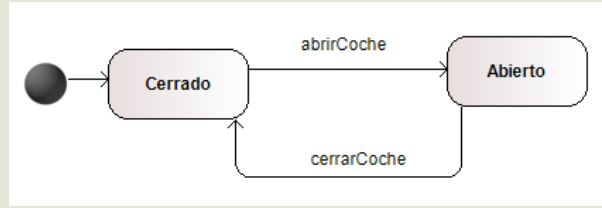
Resumen de la “metodología”



Caso de estudio
Creación de templates







Relevant methods

```
cogerLlave,SQ,cogerLlave_SQ_tmpl.provn
abrirCoche,SQ,abrirCoche_SQ_tmpl.provn
cerrarCoche,SD,cerrarCoche_SD_tmpl.provn
abrirCoche,SD,abrirCoche_SD_tmpl.provn
```

CogerLlave(coche)

```
entity(var:input0)
entity(var:output0)
used(var:message, var:input0, -, [prov:role='ex:coche'])
wasGeneratedBy(var:output0,var:message,-,[prov:role='ex:llaveCoche'] )
wasDerivedFrom(var:output0,var:input0)

activity(var:message, [prov:type = 'ex:cogerLlave',
    tmpl:startTime = 'var:messageStartTime', tmpl:endTime = 'var:messageEndTime' ])
agent(var:agent, [prov:type='ex:Persona'])
wasAttributedTo(var:message, var:agent)
```

abrirCoche(llaveCoche)

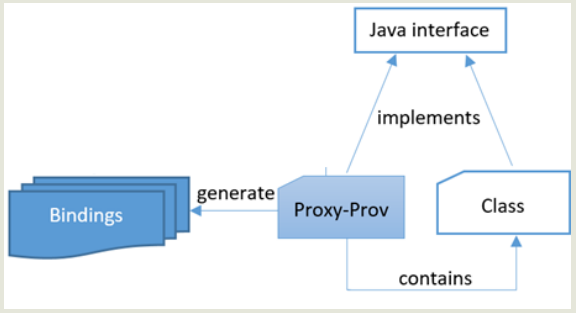
```
entity(var:input0)
used(var:message, var:input0, -, [prov:role='ex:llaveCoche'])

activity(var:message, [prov:type = 'ex:abrirCoche',
    tmpl:startTime = 'var:messageStartTime', tmpl:endTime = 'var:messageEndTime' ])
agent(var:agent, [prov:type='ex:Persona'])
wasAttributedTo(var:message, var:agent)
```

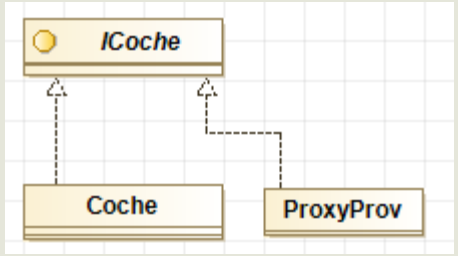
abrirCoche

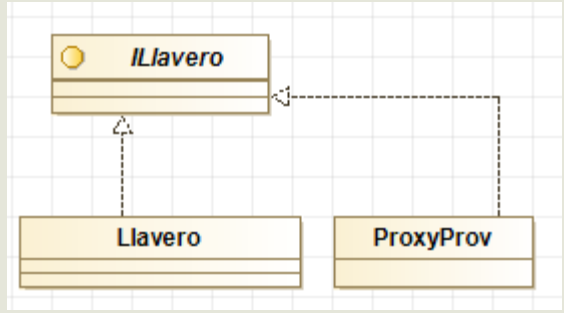
```
entity(var:ObjectModelled, [prov:type='var:ObjectModelledType'])
entity(var:target, [prov:type='var:ObjectModelledType' , ex:state= 'ex:Abierto'])
specializationOf(var:target,var:ObjectModelled)
wasGeneratedBy(var:target, var:event, -)
wasDerivedFrom(var:target,var:source)

entity(var:source, [prov:type='var:ObjectModelledType' , ex:state= 'ex:Cerrado'])
activity(var:event, [prov:type = 'ex:abrirCoche',
  tmpl:startTime = 'var:actionStartTime', tmpl:endTime = 'var:actionEndTime' ])

activity(var:event, [prov:type = 'ex:abrirCoche'])

used(var:event, var:source, -)
specializationOf(var:source,var:ObjectModelled)
```

```
wasGeneratedBy(var:target, var:action, -)
wasDerivedFrom(var:target,var:source)

entity(var:source, [prov:type='var:ObjectModelledType' , ex:state= 'ex:Abierto'])
activity(var:action, [prov:type = 'ex:cerrarCoche',
  tmpl:startTime = 'var:actionStartTime', tmpl:endTime = 'var:actionEndTime' ])

activity(var:event, [prov:type = 'ex:cerrarCoche'])
wasInformedBy(var:event, var:action)

used(var:action, var:source, -)
specializationOf(var:source,var:ObjectModelled)
```

Caso de estudio
Generación de bindings








```
entity(var:ObjectModelled,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21',
    tmpl:2dvalue_0_0 = 'ex:Coche'])

entity(var:input0,[tmpl:value_0 = 'ex:probatinas.Coche.LLave@e6ea0c6'])

entity(var:message,[tmpl:value_0 = 'ex:abrirCoche_1_0'])
entity(var:messageStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:messageEndTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

entity(var:event,[tmpl:value_0 = 'ex:abrirCoche_1_0'])
entity(var:eventStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
```

```
document
prefix tmpl <http://openprovenance.org/tmpl#>
prefix var <http://openprovenance.org/var#>
prefix ex <http://example.org/>

entity(var:input0,[tmpl:value_0 = 'ex:probatinas.Coche.LLavero@abcdef'])
entity(var:output0,[tmpl:value_0 = 'ex:probatinas.Coche.LLave@e6ea0c6'])
entity(var:message,[tmpl:value_0 = 'ex:cogerLlave_1_0'])
entity(var:messageStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:messageEndTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:agent,[tmpl:value_0 = 'ex:probatinas.Coche.Person@54c87c91_1'])
endDocument
```

```
ICoche ferrari = (ICoche) ProxyProv.generateProxy(new Coche());
ILLavero llavero = (ILLavero) ProxyProv.generateProxy(new LLavero());

LLave llaveCoche = llavero.cogerLlave( coche: "Ferrari");

ferrari.abrirCoche(llaveCoche);
```

Caso de estudio
Generación de bindings

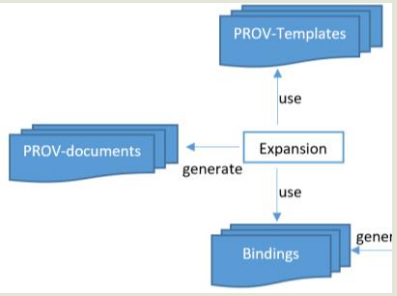
```
Coche ferrari = new Coche();
LLavero llavero = new LLavero();

LLave llaveCoche = llavero.cogerLlave( coche: "Ferrari");

ferrari.abrirCoche(llaveCoche);
```

```
ICoche ferrari = (ICoche) ProxyProv.generateProxy(new Coche());
ILLavero llavero = (ILLavero) ProxyProv.generateProxy(new LLavero());

LLave llaveCoche = llavero.cogerLlave( coche: "Ferrari");

ferrari.abrirCoche(llaveCoche);
```

Caso de estudio
Generación de bindings

```
////////////////////////////////   activityEntity    ////////////////////////////////
Entity activityEntity = pfactory.newEntity(var( param: "message"));
activityEntity.getOther().add(pfactory.newOther(tmpl( param: "value_0"), ex( param: m.getName() + "_" + generateIdentifier(m.getName()) + "_" + getC
listEntities.add(activityEntity);
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//Invocation the method capturing start and end Date
SimpleDateFormat xsdDateTime = new SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:ss");
//start It is important 2dvalue_0_0 when the variable is an attribute
Entity startTimeEntity = pfactory.newEntity(var( param: "messageStartTime"));
startTimeEntity.getOther().add(pfactory.newOther(tmpl( param: "2dvalue_0_0"), xsdDateTime.format(start), pfactory.getName().XSD_DATETIME));
listEntities.add(startTimeEntity);

//end It is important 2dvalue_0_0 when the variable is an attribute
Entity endTimeEntity = pfactory.newEntity(var( param: "messageEndTime"));
endTimeEntity.getOther().add(pfactory.newOther(tmpl( param: "2dvalue_0_0"), xsdDateTime.format(end), pfactory.getName().XSD_DATETIME));
listEntities.add(endTimeEntity);
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Caso de estudio



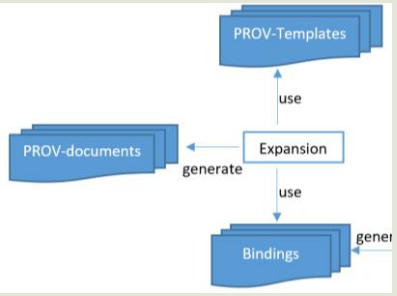

```
entity(var:ObjectModelled, [prov:type='var:ObjectModelledType'])
entity(var:target, [prov:type='var:ObjectModelledType' , ex:state= 'ex:Abierto'])
specializationOf(var:target,var:ObjectModelled)
wasGeneratedBy(var:target, var:event, -)
wasDerivedFrom(var:target,var:source)

entity(var:source, [prov:type='var:ObjectModelledType' , ex:state= 'ex:Cerrado'])
activity(var:event, [prov:type = 'ex:abrirCoche',
  tmpl:startTime = 'var:actionStartTime', tmpl:endTime = 'var:actionEndTime' ])

activity(var:event, [prov:type = 'ex:abrirCoche'])

used(var:event, var:source, -)
specializationOf(var:source,var:ObjectModelled)
```

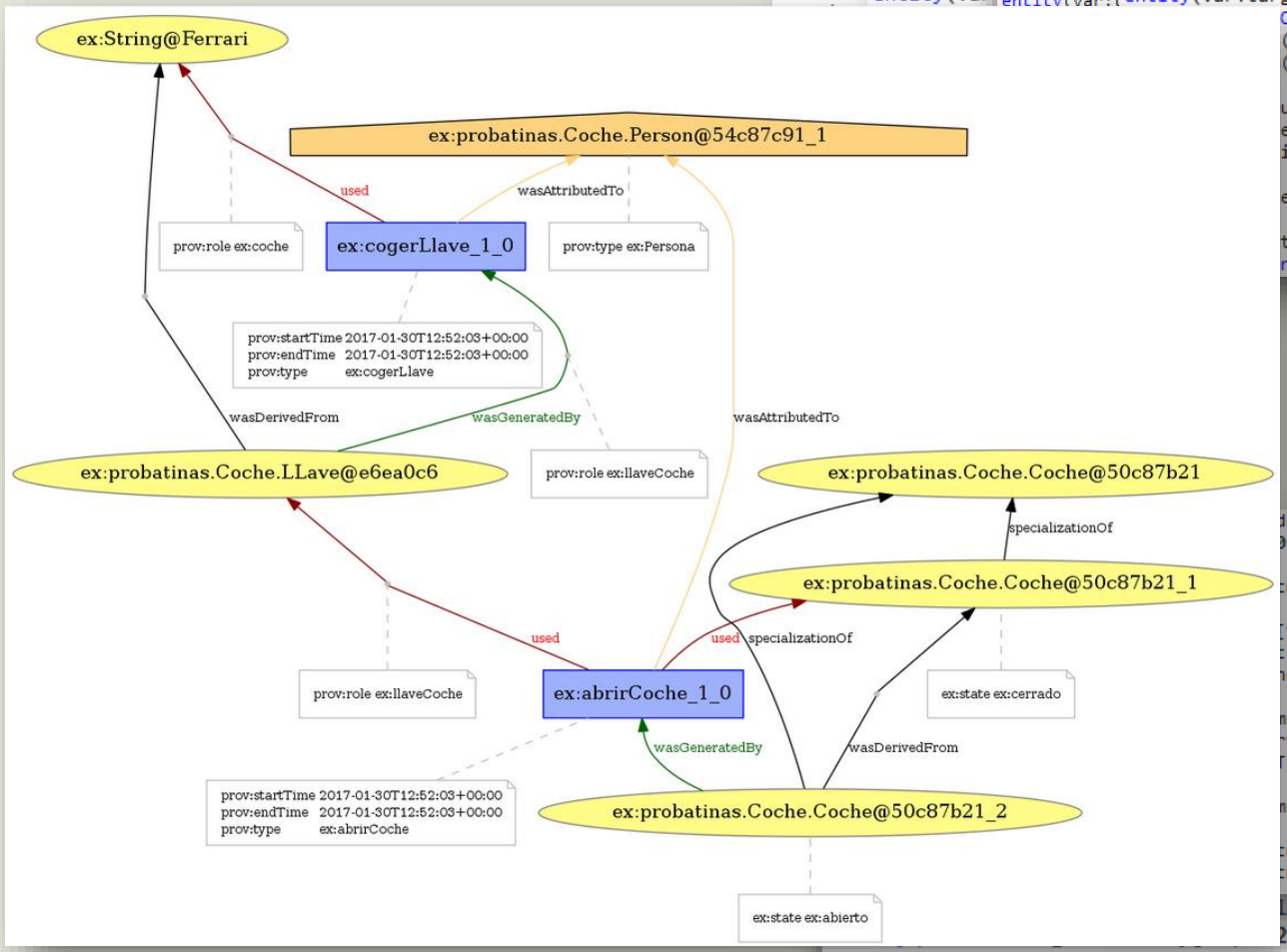
Templates

```
entity(var:ObjectModelled,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21',
    tmpl:2dvalue_0_0 = 'ex:Coche'])

entity(var:input0,[tmpl:value_0 = 'ex:probatinas.Coche.LLave@e6ea0c6'])

entity(var:message,[tmpl:value_0 = 'ex:abrirCoche_1_0'])
entity(var:messageStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:messageEndTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

entity(var:event,[tmpl:value_0 = 'ex:abrirCoche_1_0'])
entity(var:eventStartTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:eventEndTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

entity(var:agent,[tmpl:value_0 = 'ex:probatinas.Coche.Person@54c87c91_1'])

entity(var:source,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21_1'])
entity(var:target,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21_2'])
```

Bindings

Caso de estudio






```
entity(var:ObjectModelled, [prov:type='var:ObjectModelledType'])
entity(var:target, [prov:type='var:ObjectModelledType' , ex:state= 'ex:Abierto'])
Of(var:target,var:ObjectModelled)
(var:target, var:event, -)
(var:target,var:source)

urce, [prov:type='var:ObjectModelledType' , ex:state= 'ex:Cerrado'])
event, [prov:type = 'ex:abrirCoche',
ime = 'var:actionStartTime', tmpl:endTime = 'var:actionEndTime' ])

event, [prov:type = 'ex:abrirCoche'])

t, var:source, -)
nOf(var:source,var:ObjectModelled)
```

```
delled,[tmpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21',
) = 'ex:Coche'])

mpl:value_0 = 'ex:probatinas.Coche.LLave@e6ea0c6'])

[tmpl:value_0 = 'ex:abrirCoche_1_0'])
artTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
dTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

pl:value_0 = 'ex:abrirCoche_1_0'])
tTime,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
Time,[tmpl:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])

pl:value_0 = 'ex:probatinas.Coche.Person@54c87c91_1'])

mpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21_1'])
mpl:value_0 = 'ex:probatinas.Coche.Coche@50c87b21_2'])

l:2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
2dvalue_0_0 = "2017-01-30T13:52:03" %% xsd:dateTime])
entity(var:agent,[tmpl:value_0 = 'ex:probatinas.Coche.Person@54c87c91_1'])
endDocument
```

Bibliografía

Luc Moreau https://lucmoreau.wordpress.com/

Dong Huynh http://trungdong.github.io/

Provenance

The Foundations for Provenance on the Web. Luc Moreau
Lineage Retrieval for Scientific Data Processing. A survey. R Bose, J Frew
Provenance and scientific workflows: challenges and opportunities. SB Davidson, J Freire

PROV

PROV-OVERVIEW. Descripción general de la familia de documentos PROV.
PROV-PRIMER. Manual básico del modelo de datos PROV.
PROV-DM. El modelo de datos PROV para provenance.
Provenance: An Introduction to PROV. Luc Moreau, Paul Groth
The rationale of PROV. Luc Moreau, Paul Groth , James Cheney, Timothy Lebo, Simon Miles

Preguntas, discusión, sugerencias, trabajo futuro…