

# Creación de DSLs gráficos con Sirius

## Jornadas sobre Ingeniería del Software y Bases de Datos

Alberto Hernández Chillón

alberto.hernandez1@um.es

Cátedra SAES-UMU  
Universidad de Murcia



Jesús García Molina

jmolina@um.es

Facultad de Informática  
Universidad de Murcia

Tenerife, España, Julio 2017




UNIVERSIDAD DE  
MURCIA


# Presentación y material




<https://github.com/Soltari/>


 Search GitHub

Pull requestsIssuesMarketplaceGist



**Alberto Hernández**  
Soltari  
[Add a bio](#)

 University of Murcia  
Murcia, Spain  
[alberto.hernandez1@um.es](mailto:alberto.hernandez1@um.es)

**Organizations**  


OverviewRepositories 4Stars 3Followers 3Following 4

Popular repositoriesCustomize your pinned repositories

**NoSQLVisualizationTools**  
NoSQL Schema and data visualization tools  
Java 3

**UI\_Splash\_Screen**  
First commit test  
Java

**Utils**  
Java translation service  
Java

**Tutorial\_Sirius**  
Java


235 contributions in the last yearContribution settings ▾

	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul
Mon													
Wed													
Fri													

[Learn how we count contributions.](#) Less More

Contribution activityJump to ▾ 2017

July 2017

 Created 17 commits in 1 repository  
[catedrasaes-umu/nosql](#) 17 commits

[Show more activity](#)

2016  
2015  
2014  
2013

# Índice de contenido

- 1 Introducción a Sirius
- 2 Instalación de Sirius y componentes
- 3 Aspectos básicos de creación de DSLs con Sirius
- 4 Desarrollo de un caso práctico con Sirius
- 5 Aspectos avanzados de creación de DSLs con Sirius
- 6 Distribución del DSL gráfico
- 7 Consideraciones y valoraciones
- 8 Referencias y material de consulta

# Índice de contenido

- 1 Introducción a Sirius
- 2 Instalación de Sirius y componentes
- 3 Aspectos básicos de creación de DSLs con Sirius
- 4 Desarrollo de un caso práctico con Sirius
- 5 Aspectos avanzados de creación de DSLs con Sirius
- 6 Distribución del DSL gráfico
- 7 Consideraciones y valoraciones
- 8 Referencias y material de consulta



## The easiest way to get your own Modeling Tool!

- *Visual*: Diagrams, tables and trees
- *Declarative*: No code generation
- *Easy*: Your modeling workbench in hours
- Reduce the Tooling Learning Curve
- Decrease the cost of your tools
- Documentation, Forum, Professional Support

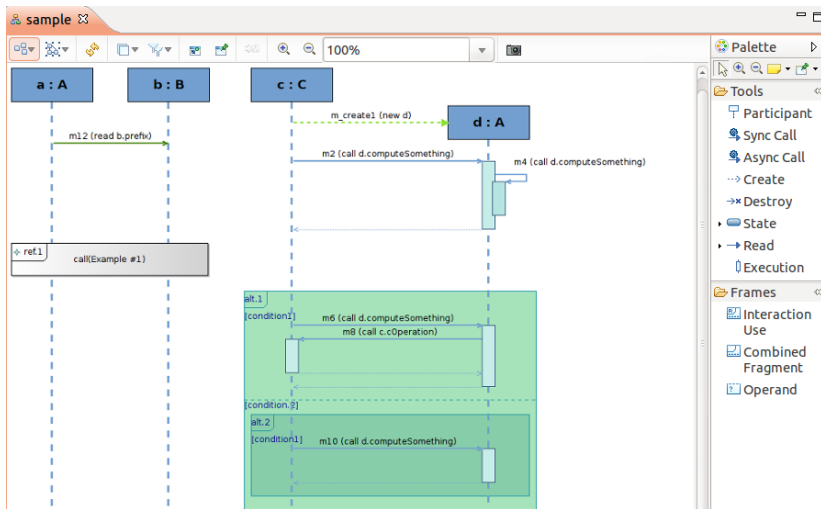
(<http://eclipse.org/sirius/>)

- Desarrollado por Obeo como herramienta interna para Thales
- Demo mostrada durante el evento *EclipseCon France 2013*
- Buena recepción y apoyo de la comunidad de Eclipse
  - *EclipseCon*, *SiriusCon*, conferencias de modelado...
  - Constante demanda de nueva funcionalidad
- Actualizaciones y soporte activo
  - Sirius 2.x: Actualizaciones en enero de 2017
  - Sirius 3.x: Actualizaciones en abril de 2017
- Última versión disponible: Sirius 4.1.5 el 15 de junio de 2017
- **Edit:** ¡Sirius 5.0.0 el 28 de junio de 2017!

## Sirius se basa en los siguientes puntos:

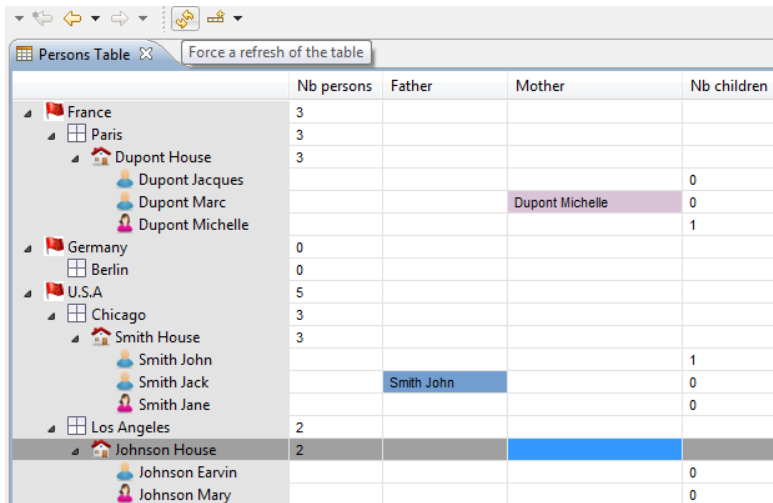
- Definición de una serie de vistas (*viewpoints*)
- Sin generación de código asociado
- Para cada vista el desarrollador determina:
  - La asociación entre cada elemento del metamodelo y su representación
  - El aspecto gráfico y estilo de cada elemento
  - Los elementos que puede crear el usuario
  - Las reglas de validación del modelo
- Generación de un editor para crear, visualizar y manipular modelos
- El usuario final crea y manipula modelos con estas vistas

# DSLs gráficos - Secuencias





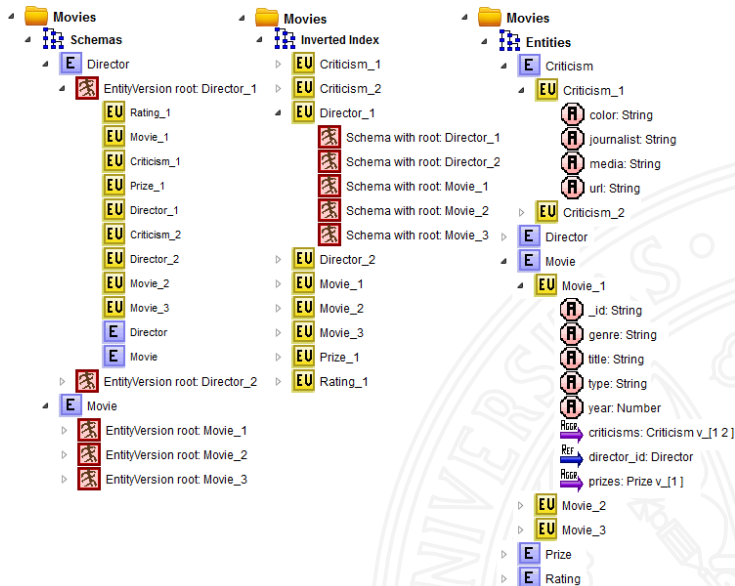
# DSLs gráficos - Tablas



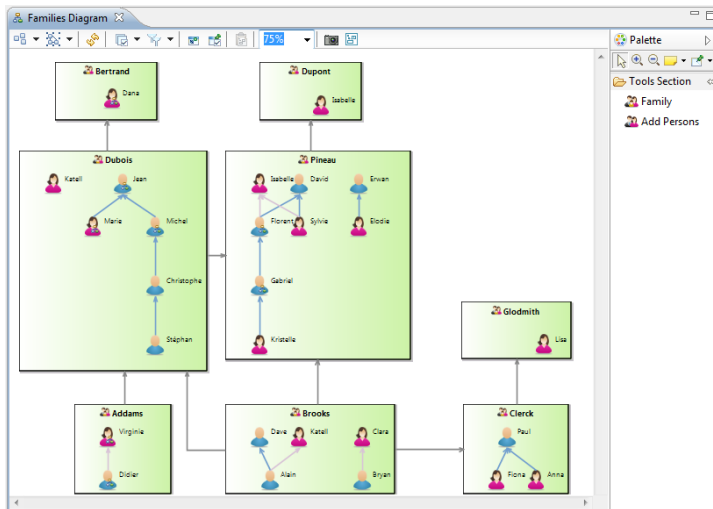
Persons Table Force a refresh of the table

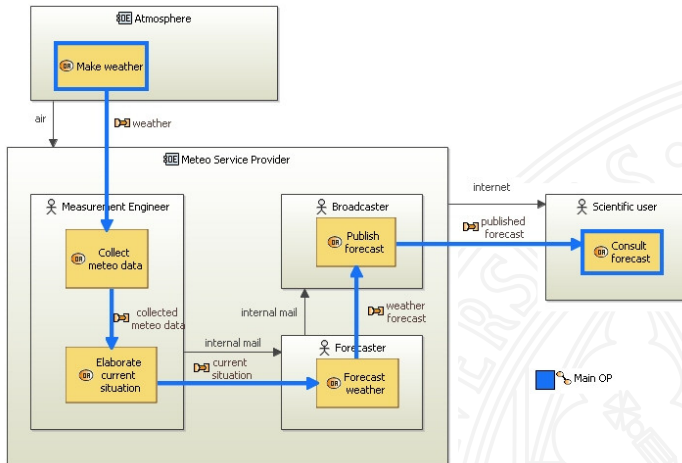
	Nb persons	Father	Mother	Nb children
France	3			
Paris	3			
Dupont House	3			
Dupont Jacques				0
Dupont Marc			Dupont Michelle	0
Dupont Michelle				1
Germany	0			
Berlin	0			
U.S.A	5			
Chicago	3			
Smith House	3			
Smith John				1
Smith Jack		Smith John		0
Smith Jane				0
Los Angeles	2			
Johnson House	2			
Johnson Earvin				0
Johnson Mary				0

# DSLs gráficos - Árboles



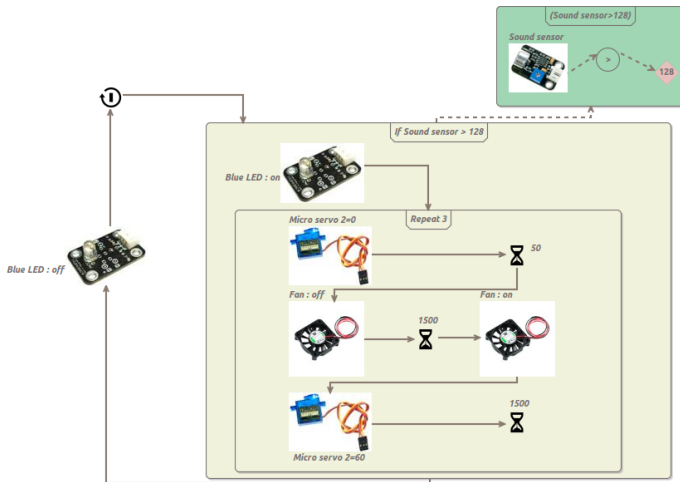
# DSLs gráficos - Diagramas







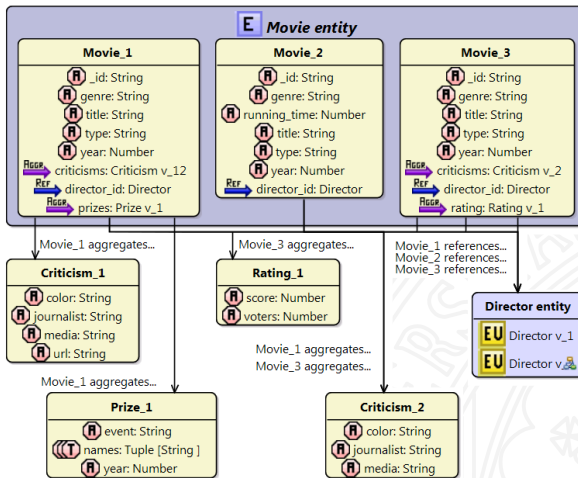
Arduino designer



# DSLs gráficos - Galería (III)




## NoSQL Visualization Tool





# Índice de contenido


- 1 Introducción a Sirius
- 2 Instalación de Sirius y componentes**
- 3 Aspectos básicos de creación de DSLs con Sirius
- 4 Desarrollo de un caso práctico con Sirius
- 5 Aspectos avanzados de creación de DSLs con Sirius
- 6 Distribución del DSL gráfico
- 7 Consideraciones y valoraciones
- 8 Referencias y material de consulta

# Instalación de Sirius y componentes (I)

  
Ready-to-Use Package

  
Drag to Install

  
Marketplace

  
Update Site

## Download a Ready-to-Use Package

This free package has been created by Sirius committers to facilitate your first steps with Sirius. It contains Sirius, neatly integrated with other Open Source technologies (EMF Compare, eGit and SWTBot).

[DOWNLOAD OBEO DESIGNER COMMUNITY](#)

## Need help to deploy Sirius?

Obeo, co-leader of Sirius, provides professional services from the set-up to the deployment of your industrial-strength modeling workbenches created with Sirius.

[Training](#) | [Consulting & Coaching](#) | [Custom Development](#) | [Support](#) ➔

## Need collaborative features for Sirius?

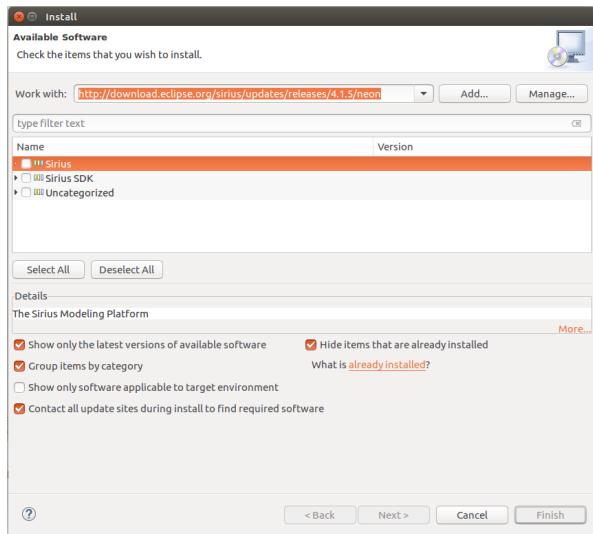
Obeo Designer Team edition facilitates the collaboration with your other team members by storing your models and representations (diagrams, tables, trees) created with Sirius in a shared repository.

[Read more](#) ➔

(<http://www.eclipse.org/sirius/download.html>)



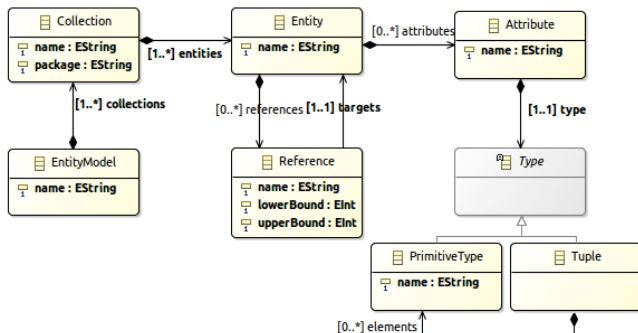
# Instalación de Sirius y componentes (II)



# Índice de contenido

- 1 Introducción a Sirius
- 2 Instalación de Sirius y componentes
- 3 Aspectos básicos de creación de DSLs con Sirius**
- 4 Desarrollo de un caso práctico con Sirius
- 5 Aspectos avanzados de creación de DSLs con Sirius
- 6 Distribución del DSL gráfico
- 7 Consideraciones y valoraciones
- 8 Referencias y material de consulta

# Metamodelo y modelo iniciales



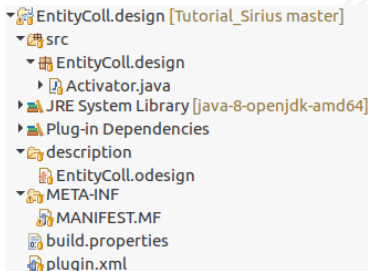
- ✦ Entity Model Stuff
  - ✦ Collection Media
    - ✦ Entity Social network
    - ✦ Entity Newspaper
    - ✦ Entity Radio
    - ✦ Entity TV
    - ✦ Entity Magazine
  - ✦ Collection Books
    - ✦ Entity Book
    - ✦ Entity Publisher
    - ✦ Entity Author
    - ✦ Entity Journal
    - ✦ Entity Company
    - ✦ Entity Content
  - ✦ Collection Restaurants
    - ✦ Entity Restaurant
    - ✦ Entity Waiter
    - ✦ Entity Table
    - ✦ Entity Menu
    - ✦ Entity Dish
    - ✦ Entity Ingredient

## Escenario de partida:

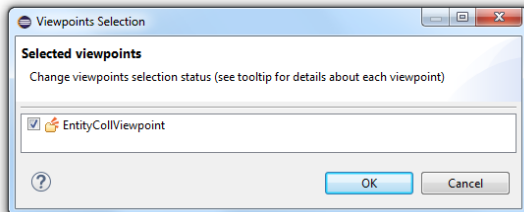
- Se dispone de un metamodelo instalado en Eclipse
- Se dispone de la última versión de Sirius

## File ⇒ New ⇒ Viewpoint Specification Project:

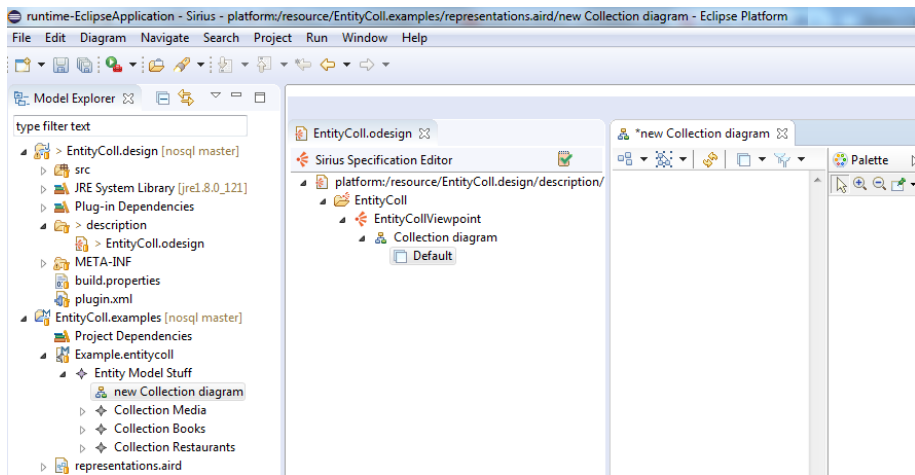
- Implementado a partir de un *Eclipse plug-in project*



- En el fichero *odesign* se almacenan los distintos *viewpoints*
- Cada *viewpoint* contiene distintas representaciones: *diagramas*, *árboles*, *secuencias*...
- Cada representación tiene asociado un metamodelo y un elemento raíz
- Se pueden asociar *viewpoints* sobre un proyecto Eclipse con modelos



# Creando la primera representación en Sirius



**Cada elemento se puede representar de las siguientes formas:**

- *Contenedor*: Para elementos complejos y con anidación
- *Nodo*: Para elementos simples. No permite anidación
- *Arco basado en elemento*: Elementos que actúan de relación
- *Arco basado en relación*: Para visualizar relaciones entre entidades

**Para cada *contenedor* y *nodo* se debe especificar:**

- Identificador único
- Asociación con un elemento del metamodelo
- Candidatos semánticos a representar: Cuáles de los elementos del metamodelo deben representarse
- Estilo a aplicar en la representación visual

# Creando elementos visuales en Sirius

The screenshot displays the Sirius IDE interface. The top-left pane shows the 'Sirius Specification Editor' with a tree view of the project structure: 'platform:/resource/EntityColl.design/description/EntityColl.odesign' > 'EntityColl' > 'EntityCollViewpoint' > 'Collection diagram' > 'Default' > 'Collection'. The top-right pane shows a 'new Collection diagram' with a toolbar and a diagram area. The bottom pane is the 'Properties' view, currently showing the 'Container' tab. The 'General' section is active, displaying the following properties:

- Id\*:** Collection
- Label:** Collection
- Domain Class\*:** entityColl.Collection
- Semantic Candidates Expression:** aql: self.collections
- Children Presentation\*:** Free Form (selected), List, Horizontal Stack, Vertical Stack



# Creando elementos visuales en Sirius

The screenshot shows the Sirius IDE interface with the following components:

- Project Explorer:** Shows the project structure for `EntityColl.odesign`. The `EntityColl` package contains an `EntityCollViewpoint`, which in turn contains a `Collection diagram`. The `Collection diagram` package contains a `Default` package, which contains a `Collection` element.
- Diagram Editor:** Displays a new `Collection diagram`. The toolbar includes icons for creating, deleting, and editing diagram elements.
- Properties Panel:** Shows the configuration for the `Collection` element. The `Id*` field is set to `Collection`, and the `Label` field is set to `Collection`. The `Domain Class*` field is set to `entityColl.Collection`. The `Semantic Candidates Expression` field is set to `aqi: self.collections`. The `Children Presentation*` field is set to `Free Form`.

Several question marks are overlaid on the image to highlight areas of confusion or uncertainty:

- Two green question marks are placed over the `Collection diagram` package in the Project Explorer and the `Collection diagram` toolbar.
- A blue question mark is placed over the `Collection` element in the Project Explorer.
- A red question mark is placed over the `Children Presentation*` field in the Properties Panel.
- A blue question mark is placed over the `Semantic Candidates Expression` field in the Properties Panel.

## Sirius permite definir expresiones con distintos lenguajes:

- *feature*: Para acceder a miembros de un objeto  
feature: self.name
- *service*: Llamadas a métodos definidos en Java  
service: myFunction()
- **Acceleo Query Language (AQL)**: Lenguaje recomendado  
aql: self.name
  - Mezcla de Acceleo y OCL
  - Poca sobrecarga de procesamiento
  - Autocompletado, cierta inferencia de tipos y validación
  - No requiere compilación
  - *Autoflatten*: Eliminación de las *listas de listas*
  - *collect*, *filter*, *select*, *reject*...

(<https://www.eclipse.org/acceleo/documentation/aql.html>)

## Sirius requiere expresiones AQL en distintos casos:

- Aplicación de estilos condicionales:

```
aql: self.oclIsTypeOf(EntityColl.PrimitiveType)
```

- Valores de etiquetas:

```
aql: 'Nombre: ' + self.name
```

- Declaración de candidatos semánticos:

```
aql: self.collections
```

- Operaciones sobre colecciones:

```
aql: self.entities->select(e |  
e.attributes.types->filter(EntityColl::PrimitiveType)->size() > 0)
```

# Creando entidades visuales en Sirius

The screenshot shows the Sirius Specification Editor interface. The top-left pane displays the project structure:

- platform:/resource/EntityColl.design/description/EntityColl.odesign
  - EntityColl
    - EntityCollViewpoint
      - Collection diagram
        - Default
          - Collection

The top-right pane shows a diagram titled "\*new Collection diagram". The toolbar contains various diagramming tools. Two green question marks are placed over the diagram type and the diagram itself.

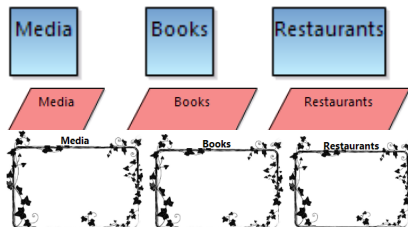
The bottom pane shows the "Properties" view for the selected "Collection" element. The "Children Presentation\*" section is highlighted with a red question mark:

- Id\*:** Collection
- Label:** Collection
- Domain Class\*:** entityColl.Collection
- Semantic Candidates Expression:** aql: self.collections
- Children Presentation\*:** Free Form (selected), List, Horizontal Stack, Vertical Stack

# Estilos aplicables en Sirius a contenedores y nodos

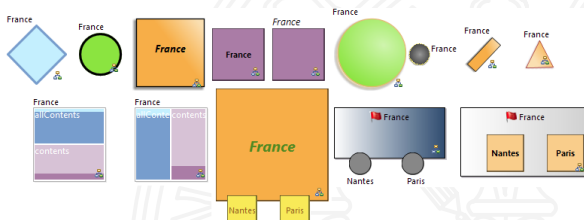
A cada contenedor se le pueden aplicar los siguientes estilos:

- *Gradient*
- *Parallelogram*
- *Workspace image*



A cada nodo se le pueden aplicar los siguientes estilos:

- *Workspace image*
- *Formas geométricas*

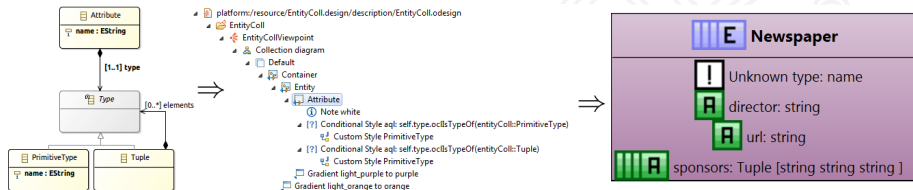


## Es posible modificar muchos aspectos de los estilos:

- Etiqueta: Mensaje, tamaño, tipo y color de letra, icono, visibilidad
- Color de frente, color de fondo, forma, borde, tamaño, *tooltip*

## Solo se puede aplicar un estilo fijo para cada elemento:

- Idea: Fijar un estilo para el caso general e implementar estilos condicionales para casos específicos
- Útil para jerarquías de metaclasses, pero no siempre es lo correcto...



# Creando entidades visuales en Sirius

The screenshot displays the Sirius Specification Editor interface. The left sidebar shows the project structure for 'EntityColl.odesign', with the 'Collection' entity selected under 'EntityCollViewpoint' > 'Collection diagram' > 'Default'. The main editor area shows a 'new Collection diagram' with a toolbar and a diagram canvas. The bottom panel is divided into 'Properties' and 'Container' tabs. The 'Container' tab is active, showing the configuration for the 'Collection' entity. The 'General' section is expanded, and the 'Children Presentation\*' property is highlighted with a red question mark. The configuration for 'Children Presentation\*' includes a 'Free Form' radio button selected, and a large red question mark is placed next to the presentation options.

**EntityColl.odesign**

- platform:/resource/EntityColl.design/description/EntityColl.odesign
  - EntityColl
    - EntityCollViewpoint
      - Collection diagram
        - Default
          - Collection

**\*new Collection diagram**

**Properties** < > Interpreter Problems Progress Console

**Container**

**General**

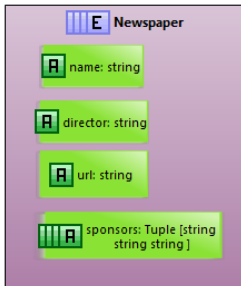
**Id\*:** Collection Label: Collection

**Domain Class\*:** entityColl.Collection

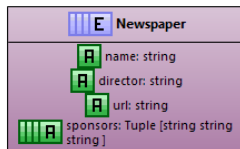
**Semantic Candidates Expression:** aql: self.collections

**Children Presentation\*:** Free Form List Horizontal Stack Vertical Stack

# Tipos de layouts para contenedores



1. Free form layout



2. List layout



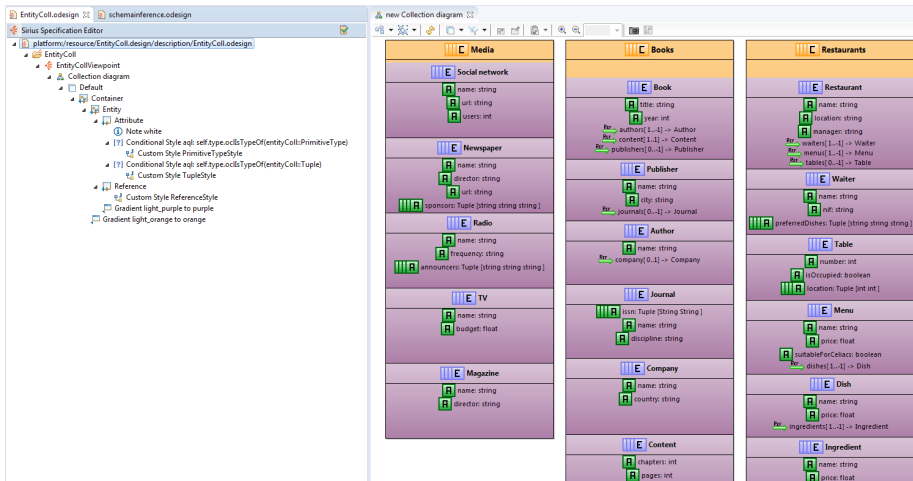
3. Horizontal stack layout



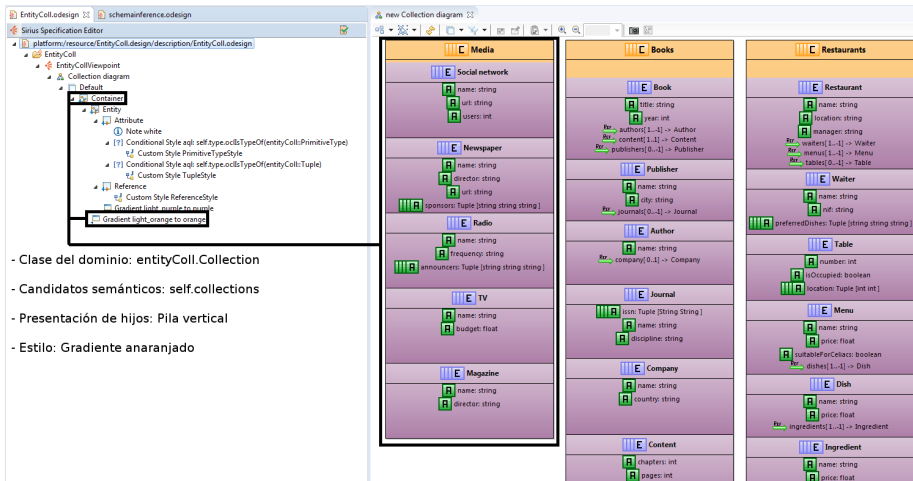
4. Vertical stack layout



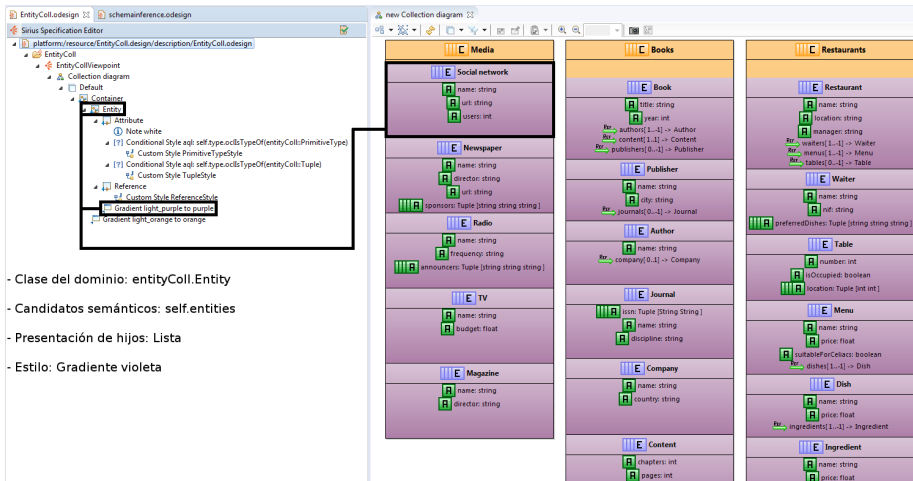
# Aplicación de los conceptos básicos



# Aplicación de los conceptos básicos

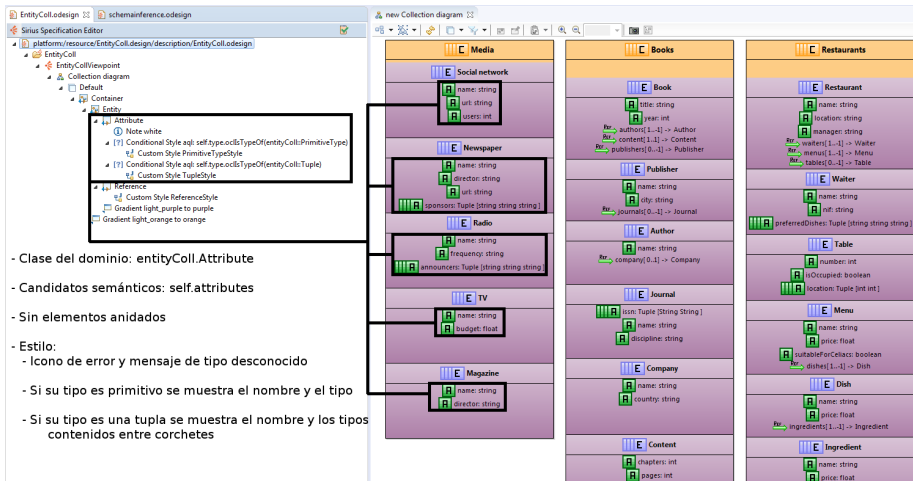


# Aplicación de los conceptos básicos

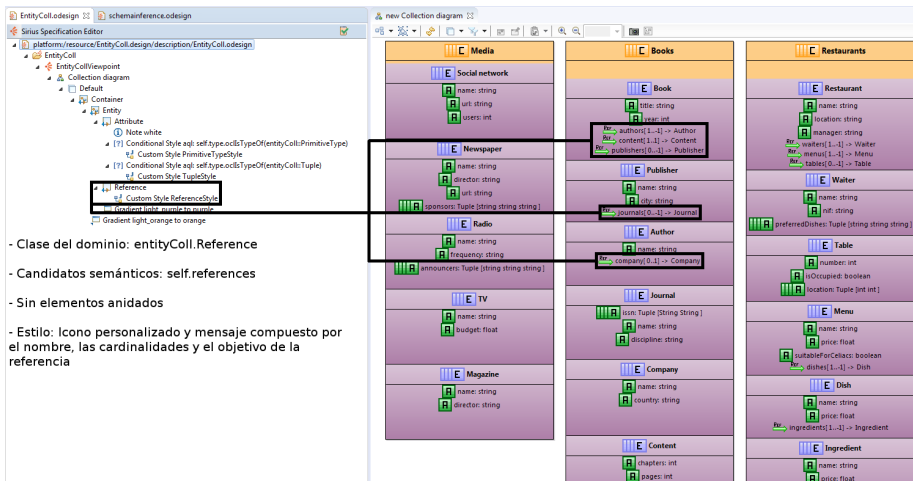


- Clase del dominio: entityColl.Entity
- Candidatos semánticos: self.entities
- Presentación de hijos: Lista
- Estilo: Gradiente violeta

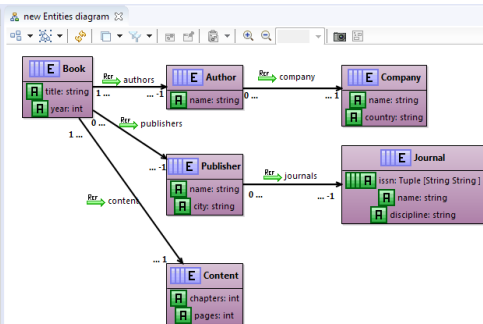
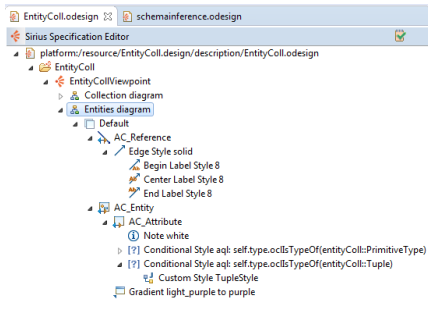
# Aplicación de los conceptos básicos



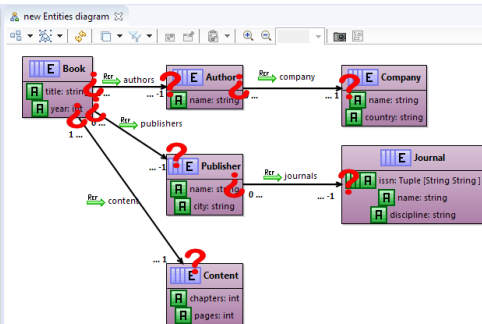
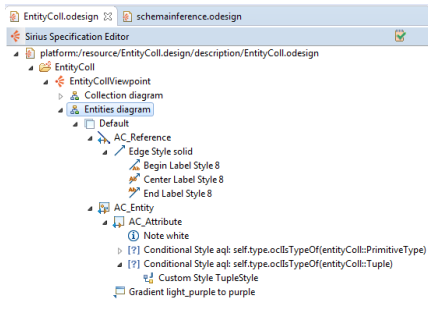
# Aplicación de los conceptos básicos



# Implementación de una vista para entidades

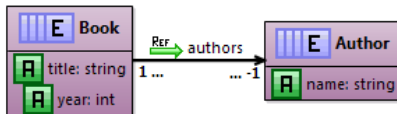


# Implementación de una vista para entidades



## Utilizados para representar relaciones entre entidades:

- Primer tipo: *Arcos basados en relación*
  - Se debe indicar la entidad origen y la entidad destino
  - Expresión de candidatos semánticos
- Segundo tipo: *Arcos basados en elemento*
  - Se debe indicar la entidad origen y la entidad destino
  - También la clase del dominio que se está representando
  - Expresiones de candidatos semánticos y de qué atributo se está mapeando



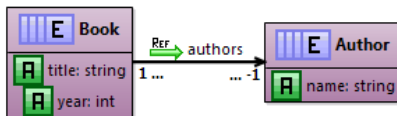


De nuevo es posible aplicar distintos estilos a los arcos:

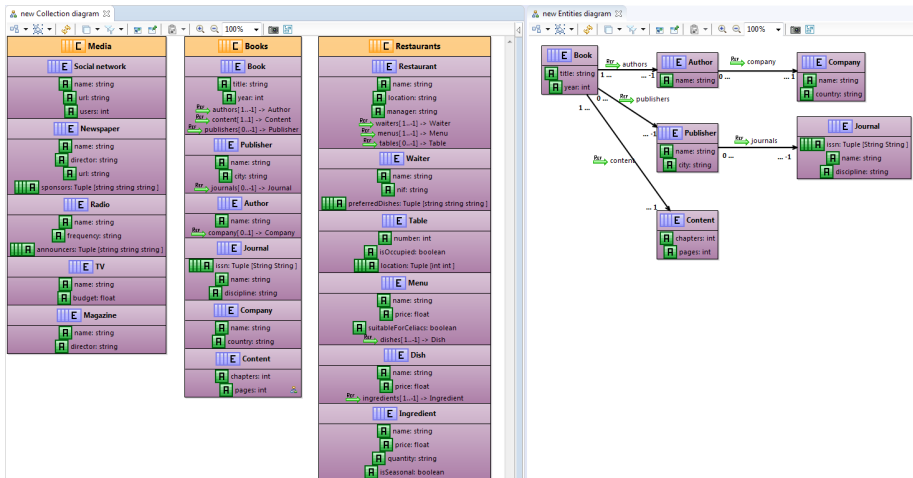
- Solidez y curvatura del trazo
- Decoradores al inicio y al final del arco
- Color, grosor, indicación de dónde debe terminar...

**Y aplicar estilos a los mensajes:**

- Mensaje, tamaño, tipo de letra, icono, color
- Se pueden colocar hasta tres mensajes al inicio, medio y fin del arco



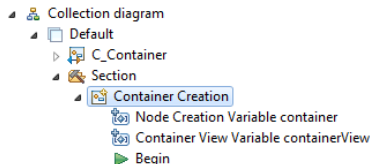
# Vistas desarrolladas hasta ahora



## Implementando la creación de elementos:

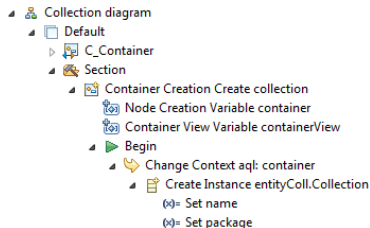
- Es posible agregar *secciones* a un diagrama
- En la sección se pueden agregar distinta funcionalidad:
  - Paleta de creación de elementos
  - Operaciones de interacción con elementos: Edición directa, reconexión de arcos, doble click, *drag & drop*...
  - Opciones de menú emergente
  - Navegación entre diagramas
  - Simulaciones, extensiones, otras secciones...¿?
- **New Tool**  $\Rightarrow$  **Section**
- **New Element Creation**  $\Rightarrow$  **Container/Node/Edge creation**

## Composición a partir operaciones genéricas:



- *Cambio de contexto, crear instancia, if, set, unset, for, switch...*
- Aplicar las operaciones correctamente es la parte complicada
- Se debe proporcionar un identificador y el nombre de una metaclassa
- Para cada operación se deben proporcionar distintos parámetros:
  - *Crear instancia*: Atributo donde agregar el objeto, qué instancia crear
  - *Set*: Nombre y valor del parámetro

# Detalle de la operación para crear colecciones



- Cambio de contexto  $\Rightarrow$  aql: container
- *Crear instancia*: Operación de creación de colecciones

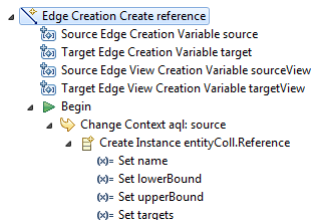
Reference name: collections, Type name: entityColl.Collections

- Set name  $\Rightarrow$  aql: 'Collection\_' +  
container.collections->size()
- Set package  $\Rightarrow$  aql: 'default'

## Paleta de la vista de colecciones

The screenshot shows the Sirius Specification Editor interface. The left pane displays the project tree for 'EntityColl.odesign', with 'EntityColl' selected. The main area shows three collection diagrams: 'Media', 'Books', and 'Restaurants'. Each diagram contains several entities, each represented by a colored box with a letter 'A' and a name. The 'Media' diagram includes 'Social network', 'Newspaper', 'Radio', and 'TV'. The 'Books' diagram includes 'Book', 'Publisher', 'Author', 'Table', 'Journal', and 'Company'. The 'Restaurants' diagram includes 'Restaurant', 'Waiter', 'Menu', 'Dish', and 'Ingredient'. The right pane shows the 'Palette' with options to create collections, entities, attributes, and references.

# Detalle de la operación para crear referencias



- *Precondición de terminación*: Un objeto no se puede autoreferenciar  
aql: `preTarget.differs(preSource)`
- Cambio de contexto  $\Rightarrow$  aql: `container`
- *Crear instancia*: Operación de creación de colecciones  
Reference name: `references`, Type name: `entityColl.Reference`
- Set targets  $\Rightarrow$  aql: `target`

# Paleta de la vista de entidades

The screenshot displays the Sirius Specification Editor interface. On the left, the 'EntityColl.odesign' file is open, showing a tree view with the following structure:

- EntityCollViewpoint
  - Collection diagram
  - Entities diagram
    - Default
      - AC\_Reference
      - AC\_Entity
      - Section Creation palette
        - Container Creation Create entity
        - Node Creation Create primitive attribute
        - Node Creation Create tuple attribute
        - Edge Creation Create reference
          - Source Edge Creation Variable source
          - Target Edge Creation Variable target
          - Source Edge View Creation Variable sourceView
          - Target Edge View Creation Variable targetView
      - Begin
        - Change Context aql: source
          - Create Instance entityColl.Reference
            - Set name
            - Set lowerBound
            - Set upperBound
            - Set targets

The main workspace shows a 'new Entities diagram' with the following entities and relationships:

- E Book**: Attributes: title: string, year: int. Relationships: 1 to ... with E Author (labeled 'Br' and 'authors'), 0 to ... with E Publisher (labeled 'Rr' and 'publishers'), 1 to ... with E Content (labeled 'Rr' and 'content').
- E Author**: Attribute: name: string. Relationship: 0 to ... with E Company (labeled 'Rr' and 'company').
- E Publisher**: Attributes: name: string, city: string. Relationship: ... to 1 with E Journal (labeled 'Rr' and 'journals').
- E Company**: Attributes: name: string, country: string.
- E Journal**: Attributes: issn: Tuple [String String], name: string, discipline: string.
- E Content**: Attributes: chapters: int, pages: int.

On the right, the 'Palette' is visible, showing the 'Creation palette' with the following options:

- Create entity
- Create primitive attribute
- Create tuple attribute
- Create reference



# Índice de contenido

- 1 Introducción a Sirius
- 2 Instalación de Sirius y componentes
- 3 Aspectos básicos de creación de DSLs con Sirius
- 4 Desarrollo de un caso práctico con Sirius**
- 5 Aspectos avanzados de creación de DSLs con Sirius
- 6 Distribución del DSL gráfico
- 7 Consideraciones y valoraciones
- 8 Referencias y material de consulta

# Índice de contenido

- 1 Introducción a Sirius
- 2 Instalación de Sirius y componentes
- 3 Aspectos básicos de creación de DSLs con Sirius
- 4 Desarrollo de un caso práctico con Sirius
- 5 Aspectos avanzados de creación de DSLs con Sirius**
- 6 Distribución del DSL gráfico
- 7 Consideraciones y valoraciones
- 8 Referencias y material de consulta

## **Hemos implementado un editor de modelos básico:**

- Elementos visuales mapeados a conceptos del metamodelo
- Personalización de estilos y condicionales
- Paleta de creación de elementos

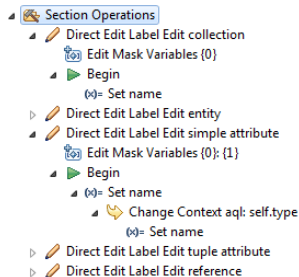
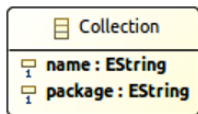
## **El editor puede enriquecerse con las siguientes utilidades:**

- Interacciones con los elementos
- Navegación entre vistas
- Filtrado de clases
- Validación del modelo
- Llamadas a servicios Java

## Tenemos un DSL gráfico...¡sin interacción!

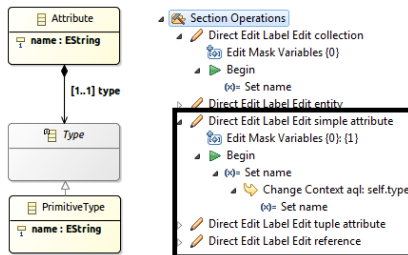
- Sirius permite crear secciones con interacciones:
  - Eliminar elementos
  - Reconectar arcos
  - Doble click sobre elementos
  - Edición de mensajes
  - *Drag & drop*
- Las interacciones se implementan haciendo uso de las operaciones genéricas:
  - *Cambio de contexto, crear instancia, if, set, unset*
  - *Move, remove, for, switch...*

## New Element Edition $\Rightarrow$ Direct edit label



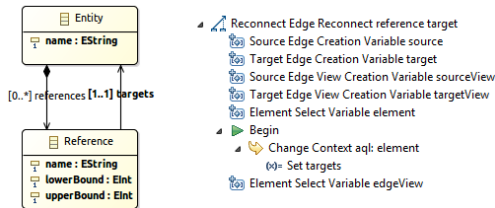
- Se debe indicar un identificador y la metaclase asociada
- Aplicar una máscara: {0} por defecto
- Set name  $\Rightarrow$  var:0

## New Element Edition $\Rightarrow$ Direct edit label



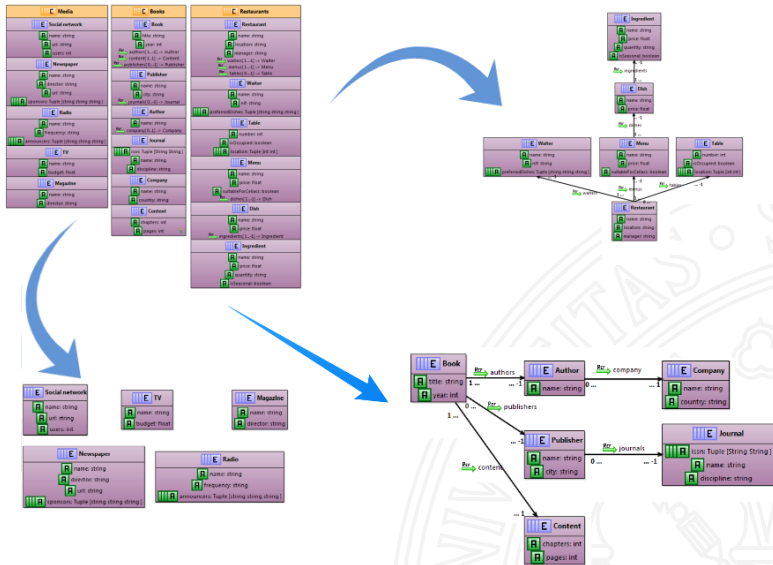
- Máscara `{0}`: `{1}`
- Set name  $\Rightarrow$  `var:0`
- *Cambio de contexto*: Para poder modificar el atributo *name* del tipo  
`aql: self.type`
- Set name  $\Rightarrow$  `var:1`

## New Element Edition $\Rightarrow$ Reconnect edge



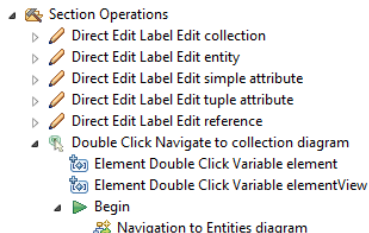
- Es posible reconectar en origen y/o objetivo de un arco
- Para reducir la complejidad se recomienda crear dos operaciones
- Variables importantes: *source*, *target*, *element*
  - *Cambio de contexto*: Para movernos a la clase *Reference*  
aql: element
  - Set targets  $\Rightarrow$  aql: target

# Navegación entre vistas





## New Element Edition $\Rightarrow$ Double click



- Tipo especial de operación asignada al doble click
- Sin problemas de sincronización
- Asignar identificador y nombre de metaclass
- Operación *navigation*:
  - Diagram description  $\Rightarrow$  Entities diagram
  - Create if not existent  $\Rightarrow$  true

## Sirius permite establecer filtros para objetos del diagrama:

- Para mostrar u ocultar elementos estáticamente o con condiciones
- **New filter**  $\Rightarrow$  **Composite filter**
- Para cada filtro se debe indicar:
  - Objetos definidos en Sirius afectados
  - Opcional: Condición de filtrado dada por una expresión
- Ejemplo: *Hide empty objects*  
`entityColl.Collection  $\Rightarrow$  aql: not(self.entities->isEmpty())`  
`entityColl.Entity  $\Rightarrow$  aql: not(self.attributes->isEmpty())`

# Aplicación de filtros

The screenshot displays the Sirius Specification Editor interface. On the left, the 'EntityColl.odesign' project is open, showing a tree view of the 'EntityColl' model. The 'Filter attributes' filter is selected. The main workspace shows a 'new Collection diagram' with a 75% zoom level. A context menu is open over the diagram, listing the following filter options:

- Filter attributes
- Filter entities
- Filter references
- Filter empty objects

The diagram contains several entities, each with its own set of attributes and relationships:

- Books**: Book (name: string, price: int)
- Restaurants**: Restaurant (name: string, location: string, manager: string, waiter: 1..\* -> Waiter, menu: 1..\* -> Menu, tables: 0..\* -> Table)
- Publisher**: Publisher (name: string, city: string)
- Author**: Author (name: string, company: 0..1 -> Company)
- Journal**: Journal (name: string, discipline: string)
- Company**: Company (name: string, country: string)
- Content**: Content (chapters: int, pages: int)
- TV**: TV (name: string, budget: float)
- Magazine**: Magazine (name: string, director: string)
- Table**: Table (number: int, isOccupied: boolean, location: Tuple (int, int))
- Menu**: Menu (name: string, price: float, suitableForCafe: boolean, dish: 1..\* -> Dish)
- Dish**: Dish (name: string, price: float, isSeasonal: boolean)
- Ingredient**: Ingredient (name: string, price: float, quantity: string, isSeasonal: boolean)

Relationships are indicated by arrows with multiplicity and directionality. For example, 'Restaurant' has a many-to-many relationship with 'Waiter' (waiter: 1..\* -> Waiter) and a many-to-many relationship with 'Menu' (menu: 1..\* -> Menu). 'Publisher' has a many-to-many relationship with 'Journal' (journal: 0..\* -> Journal). 'Author' has a many-to-many relationship with 'Journal' (journal: 0..\* -> Journal). 'Company' has a many-to-many relationship with 'Journal' (journal: 0..\* -> Journal). 'Content' has a many-to-many relationship with 'Journal' (journal: 0..\* -> Journal). 'TV' has a many-to-many relationship with 'Journal' (journal: 0..\* -> Journal). 'Magazine' has a many-to-many relationship with 'Journal' (journal: 0..\* -> Journal). 'Table' has a many-to-many relationship with 'Restaurant' (restaurant: 0..\* -> Table). 'Menu' has a many-to-many relationship with 'Restaurant' (restaurant: 0..\* -> Menu). 'Dish' has a many-to-many relationship with 'Menu' (menu: 0..\* -> Dish). 'Ingredient' has a many-to-many relationship with 'Dish' (dish: 0..\* -> Ingredient).

## Se pueden definir reglas de validación en Sirius:

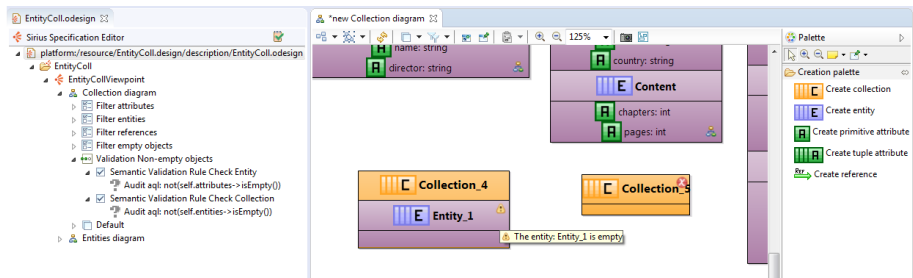
- Para asegurar la corrección del modelo formado
- Con capacidad para sugerir cambios, arreglos y gravedad del error
- **New validation**  $\Rightarrow$  **Validation**  $\Rightarrow$  **Semantic validation rule**
- Para cada regla de validación se debe indicar:
  - La importancia de la regla: *Information*, *Warning*, *Error*
  - El elemento del diagrama a analizar
  - El mensaje a mostrar
  - Una serie de condiciones a comprobar (*Audit*)
  - Opcionalmente una o varias formas de arreglar el error (*Fix*)

- Ejemplo: *Non-empty objects*

`entityColl.Collection  $\Rightarrow$  aql: not(self.entities->isEmpty())`

`entityColl.Entity  $\Rightarrow$  aql: not(self.attributes->isEmpty())`

# Aplicación de la validación



## Al diseñar un DSL gráfico podemos encontrar restricciones:

- ¿Expresar recursividad en una expresión?  $\Rightarrow$  AQL no es suficiente
- ¿Elementos sin correspondencia en el metamodelo?  $\Rightarrow$  El metamodelo no es suficiente
  - Sin tener que recurrir a transformaciones *m2m...*
- Podemos recurrir a métodos Java e invocarlos desde el DSL gráfico
- Este aspecto se ha mejorado **mucho** con las últimas actualizaciones
- **New extension**  $\Rightarrow$  **Java extension**  $\Rightarrow$  **Nombre de clase Java**
- `service: myMethod()`

# Ejemplo de utilización de servicio Java

The screenshot displays the Sirius IDE interface with three main panels:

- EntityCollServices.java**: A Java class file containing the following code:

```
package EntityColl.design.services;  
import entityColl.Entity;  
  
public class EntityCollServices  
{  
    public String getEntityTitle(Entity entity)  
    {  
        return "Brand_New_Entity_Name";  
    }  
}
```
- EntityColl.odesign**: The Sirius Specification Editor showing a tree structure of the model:
  - Validation Non-empty objects
    - Semantic Validation Rule Check Entity
      - Audit aql: not(self.attributes->isEmpty())
    - Semantic Validation Rule Check Collection
      - Audit aql: not(self.entities->isEmpty())
  - Default
    - C\_Container
      - Section Creation palette
        - Container Creation Create collection
        - Container Creation Create entity
          - Node Creation Variable container
          - Container View Variable containerView
        - Begin
          - Change Context aql: container
            - Create Instance entityColl.Entity
              - Set name
        - Node Creation Create primitive attribute
        - Node Creation Create tuple attribute
        - Node Creation Create reference
      - Section Operations
    - Entities diagram
      - EntityColl.design.services.EntityCollServices
- new Collection diagram**: A diagram showing a collection named "Collection\_5" containing an entity named "Brand\_New\_Entity\_Name".

At the bottom, the **Properties** panel is open, showing the **Set name** property for the selected element. The **Feature Name** is "name" and the **Value Expression** is "service: getEntityTitle".

# Índice de contenido

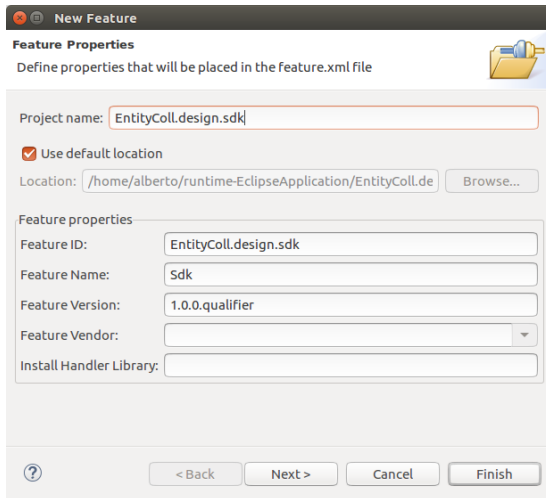
- 1 Introducción a Sirius
- 2 Instalación de Sirius y componentes
- 3 Aspectos básicos de creación de DSLs con Sirius
- 4 Desarrollo de un caso práctico con Sirius
- 5 Aspectos avanzados de creación de DSLs con Sirius
- 6 Distribución del DSL gráfico**
- 7 Consideraciones y valoraciones
- 8 Referencias y material de consulta



## Creación de un feature project:

- Se requiere que la máquina destino tenga instalado:
  - El metamodelo de partida
  - La herramienta Sirius
- Revisar el fichero *plug-in.xml*:
  - Incluir como dependencia el metamodelo base
  - Incluir en el binario carpetas de código e iconos
- **File ⇒ New ⇒ Feature project**

# Creación de un *feature project*



**New Feature**

**Feature Properties**  
Define properties that will be placed in the feature.xml file

Project name:

☒ Use default location

Location:

**Feature properties**

Feature ID:

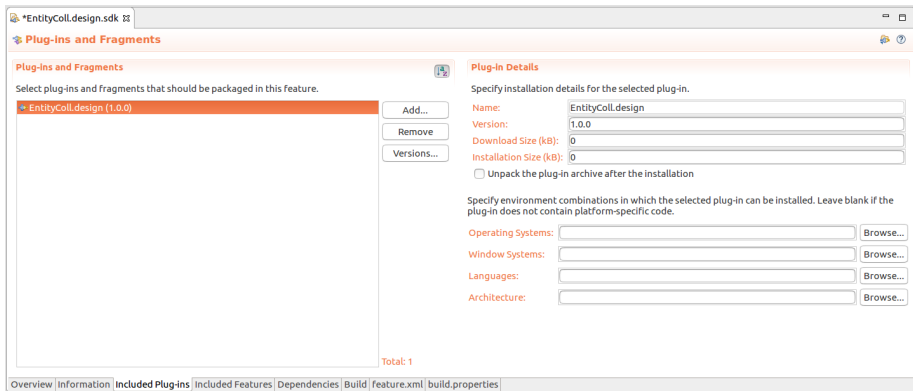
Feature Name:

Feature Version:

Feature Vendor:  ▾

Install Handler Library:

# Inclusión del proyecto *design*



# Creación de un *Update site*

File ⇒ New ⇒ Update site

**New Update Site**

**Update Site Project**  
Create a new update site project

Project name:

☒ Use default location

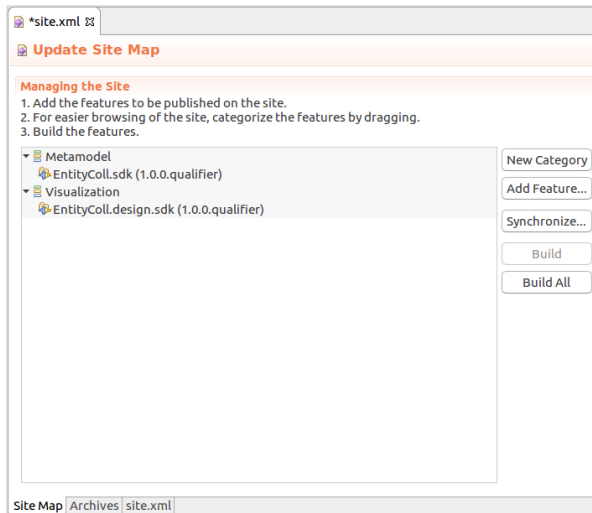
Location:

**Web Resources**

☐ Generate a web page listing all available features within the site

Web resources location:

# Inclusión del metamodelo y la visualización



# Índice de contenido

- 1 Introducción a Sirius
- 2 Instalación de Sirius y componentes
- 3 Aspectos básicos de creación de DSLs con Sirius
- 4 Desarrollo de un caso práctico con Sirius
- 5 Aspectos avanzados de creación de DSLs con Sirius
- 6 Distribución del DSL gráfico
- 7 Consideraciones y valoraciones**
- 8 Referencias y material de consulta

## Principales ventajas y beneficios:

- ✓ Generación de un editor funcional embebido en EMF
- ✓ Fácil distribución mediante *plug-ins* y *update-sites*
- ✓ Vistas y componentes altamente personalizables
- ✓ Reducción el tiempo de desarrollo del DSL gráfico
- ✓ Comunidad y soporte activos. Herramienta viva.
- ✓ Apartado de tutoriales mejorado con el tiempo:
  - Tutorial de aspectos básicos
  - Tutorial de aspectos avanzados
  - Tutorial de layouts y compartimentos
  - Tutorial de distribución del proyecto

## Aspectos mejorables, cuestiones y desventajas:











- ✗ El uso de AQL puede resultar complicado al principio
- ✗ El manual de Sirius y AQL es mejorable
- ✗ Curva de aprendizaje relativamente elevada
- ✗ Crear el primer DSL gráfico no trivial requiere esfuerzo
- ❓ Mantenimiento y evolución de la visualización y el metamodelo
- ❓ Manejo de modelos de entrada muy grandes
- ❓ Personalización del editor: Barra de herramientas y menús contextuales
  - Enriquecer el editor proporcionado no es trivial
  - Requiere manejar la mecánica de puntos de extensión en *plug-ins*



# Índice de contenido

- 1 Introducción a Sirius
- 2 Instalación de Sirius y componentes
- 3 Aspectos básicos de creación de DSLs con Sirius
- 4 Desarrollo de un caso práctico con Sirius
- 5 Aspectos avanzados de creación de DSLs con Sirius
- 6 Distribución del DSL gráfico
- 7 Consideraciones y valoraciones
- 8 Referencias y material de consulta**

# Referencias y material de consulta

-  Repositorio Git con material y vídeos
-  Tutoriales de Sirius
-  Tutorial de distribución
-  Documentación de la herramienta
  -  Manual del usuario
  -  Manual de especificación
-  Manual de mejores prácticas
-  Acceleo Query Language doc
-  Frédéric Madiot's blog
-  Cédric Brun's blog



alberto.hernandez1@um.es



<https://github.com/Soltari>