

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/Tutorial_Sirius

Alberto Hernández
Soltari
[Add a bio](#)

University of Murcia
Murcia, Spain
alberto.hernandez1@um.es

Organizations

[Overview](#) [Repositories 4](#) [Stars 3](#) [Followers 3](#) [Following 4](#)

Popular repositories [Customize 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 year [Contribution settings](#)

Contribution activity [Jump to](#) **2017**

July 2017

[Show more activity](#)

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

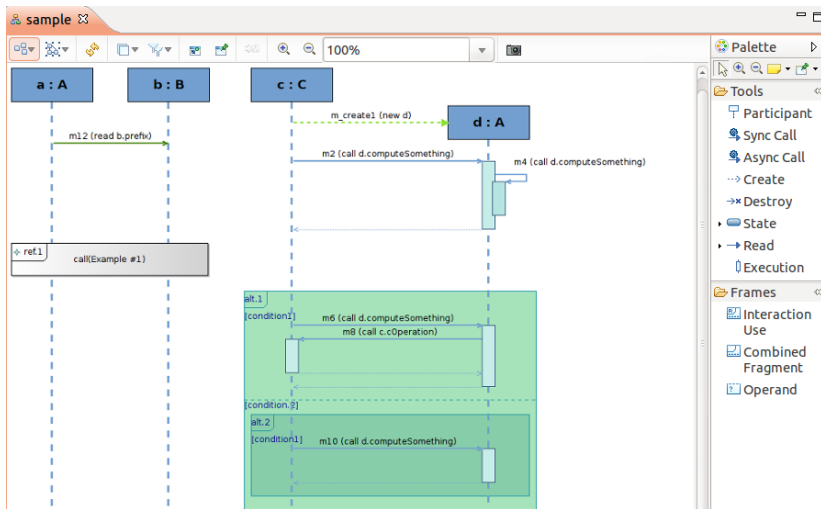
(<http://eclipse.org/sirius/>)

- 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!

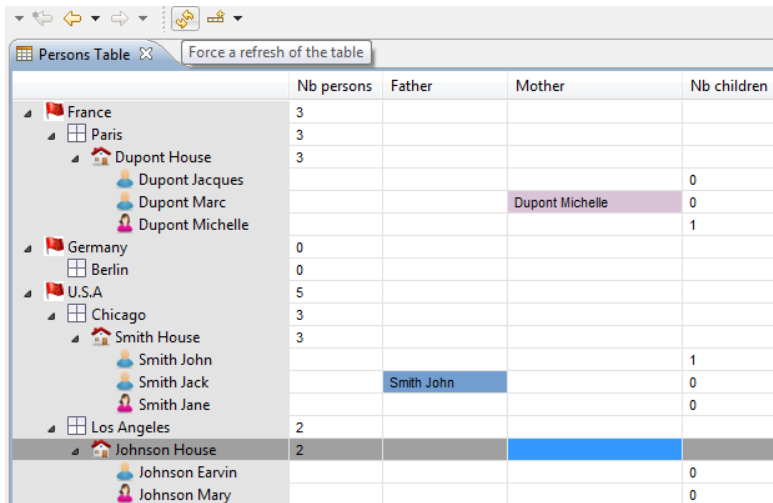
Sirius se basa en los siguientes puntos:

- 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



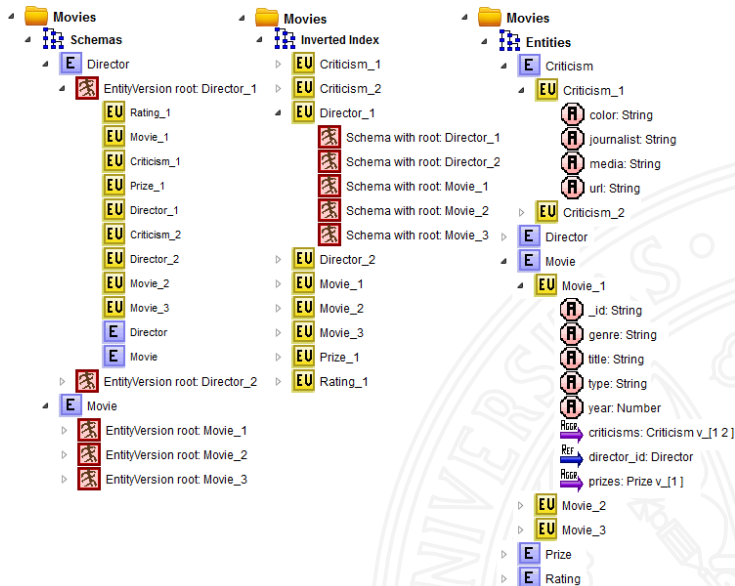
DSLs gráficos - Tablas



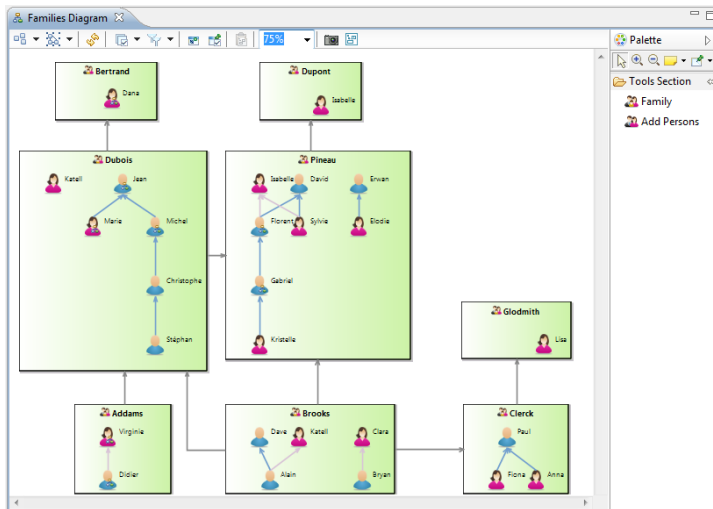
Persons Table Force a refresh of the table

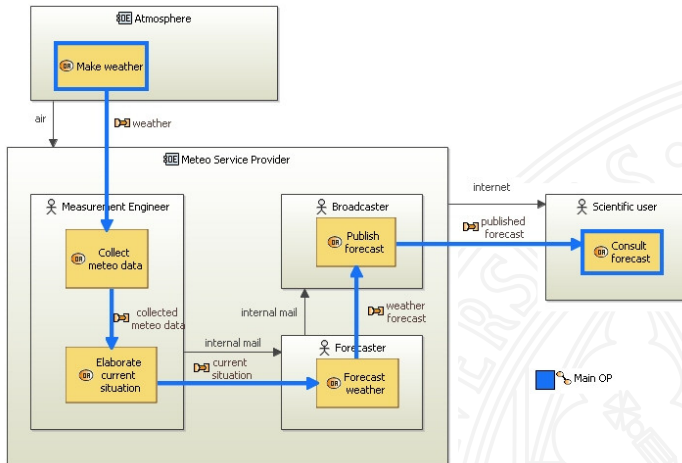
	Nb persons	Father	Mother	Nb children
France	3			
Paris	3			
Dupont House	3			
Dupont Jacques				0
Dupont Marc			Dupont Michelle	0
Dupont Michelle				1
Germany	0			
Berlin	0			
U.S.A	5			
Chicago	3			
Smith House	3			
Smith John				1
Smith Jack		Smith John		0
Smith Jane				0
Los Angeles	2			
Johnson House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

DSLs gráficos - Árboles



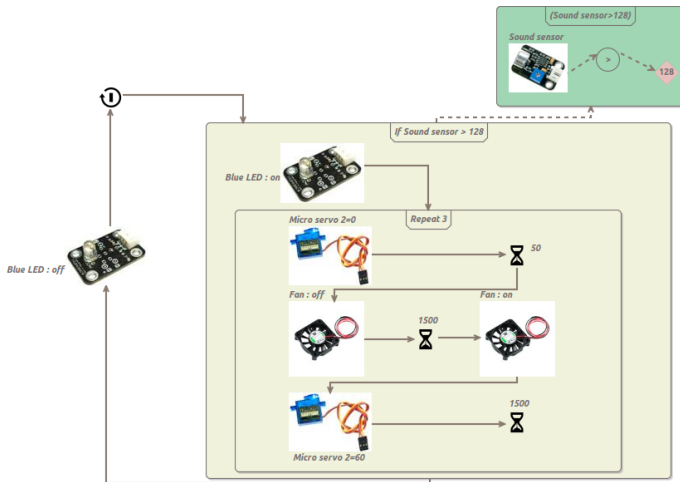
DSLs gráficos - Diagramas







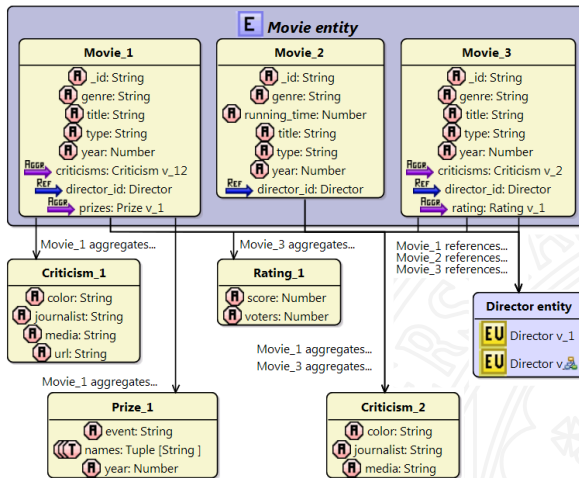
Arduino designer



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](#) ➔

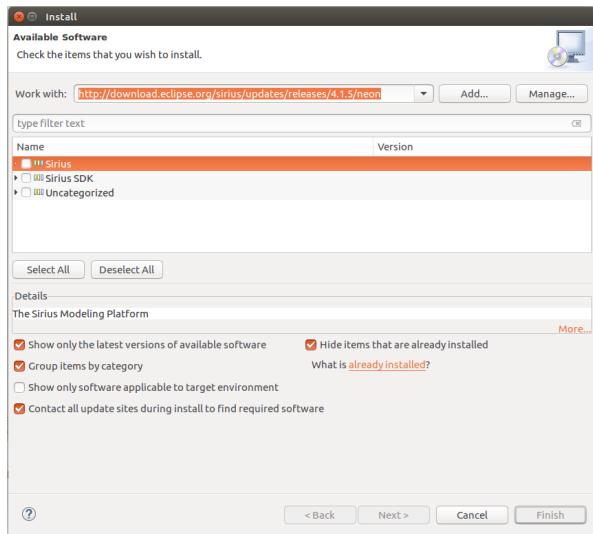
Need collaborative features for Sirius?

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>)

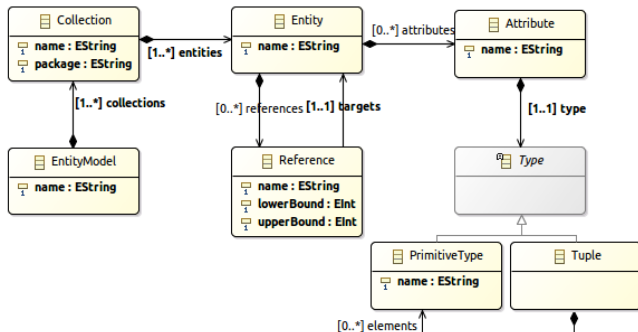
Instalación de Sirius y componentes (II)



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



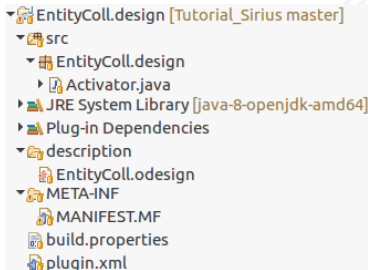
- ✦ Entity Model Stuff
 - ✦ Collection Media
 - ✦ Entity Social network
 - ✦ Entity Newspaper
 - ✦ Entity Radio
 - ✦ Entity TV
 - ✦ Entity Magazine
 - ✦ Collection Books
 - ✦ Entity Book
 - ✦ Entity Publisher
 - ✦ Entity Author
 - ✦ Entity Journal
 - ✦ Entity Company
 - ✦ Entity Content
 - ✦ Collection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

Escenario de partida:

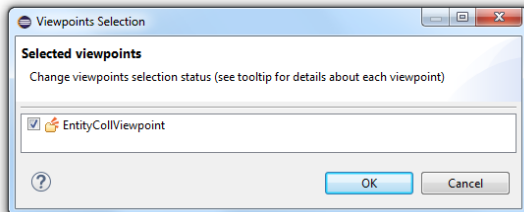
- Se dispone de un metamodelo instalado en Eclipse
- Se dispone de la última versión de Sirius

File ⇒ New ⇒ Viewpoint Specification Project:

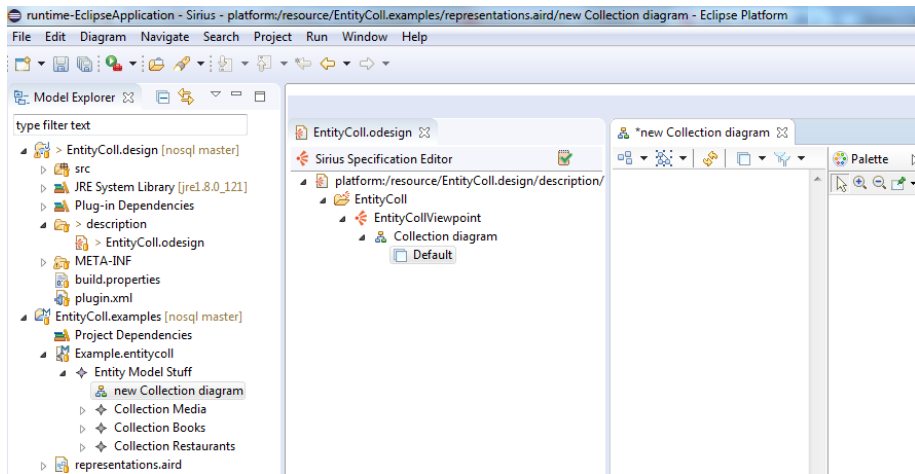
- Implementado a partir de un *Eclipse plug-in project*



- 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. It contains the following fields:

- 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. A green question mark is overlaid on the toolbar.
- Properties Panel:** Shows the configuration for the `Collection` element. The `Id*` field is set to `Collection`. The `Label` field is set to `Collection`. The `Domain Class*` field is set to `entityColl.Collection`. The `Semantic Candidates Expression` field is set to `aql: self.collections`. The `Children Presentation*` field is set to `Free Form`. A blue question mark is overlaid on the `Semantic Candidates Expression` field, and a red question mark is overlaid on the `Children Presentation*` field.

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 toolbar with various icons. Two green question marks are placed over the 'Collection diagram' icon and the 'Collection' icon.

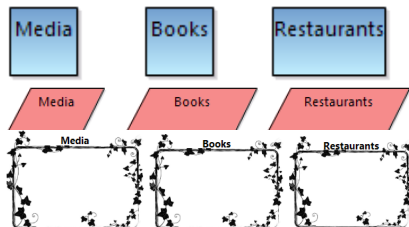
The bottom pane shows the 'Properties' view for the 'Collection' entity. The 'Children Presentation*' section is highlighted with a red question mark.

Property	Value
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

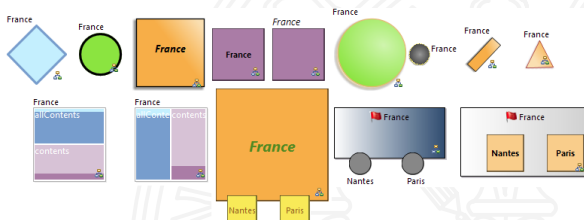
A cada contenedor se le pueden aplicar los siguientes estilos:

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



A cada nodo se le pueden aplicar los siguientes estilos:

- *Workspace image*
- *Formas geométricas*

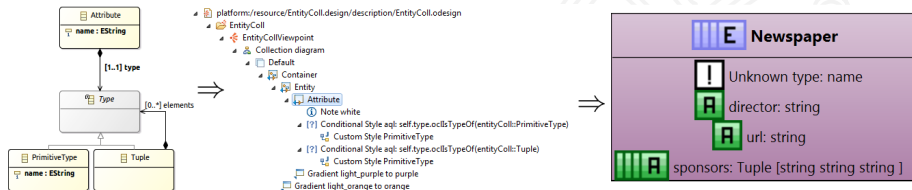


Es posible modificar muchos aspectos de los estilos:

- 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 IDE interface. On the left, the 'Sirius Specification Editor' shows a project tree for 'EntityColl.odesign' with a 'Collection' entity selected. The main workspace shows a 'new Collection diagram'. The bottom panel, titled 'Container', contains the 'Properties' view for the 'Collection' entity. The 'Children Presentation*' property is highlighted with a red question mark, and a large red question mark is placed next to the presentation options.

Properties > Interpreter Problems Progress Console

Container

General

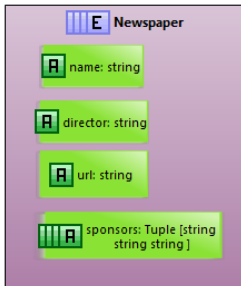
Id*: Collection Label: Collection

Domain Class*: entityColl.Collection

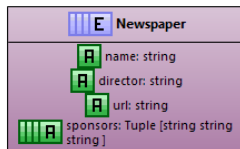
Semantic Candidates Expression: aql: self.collections

Children Presentation*: Free Form List Horizontal Stack Vertical Stack

Tipos de layouts para contenedores



1. Free form layout



2. List layout

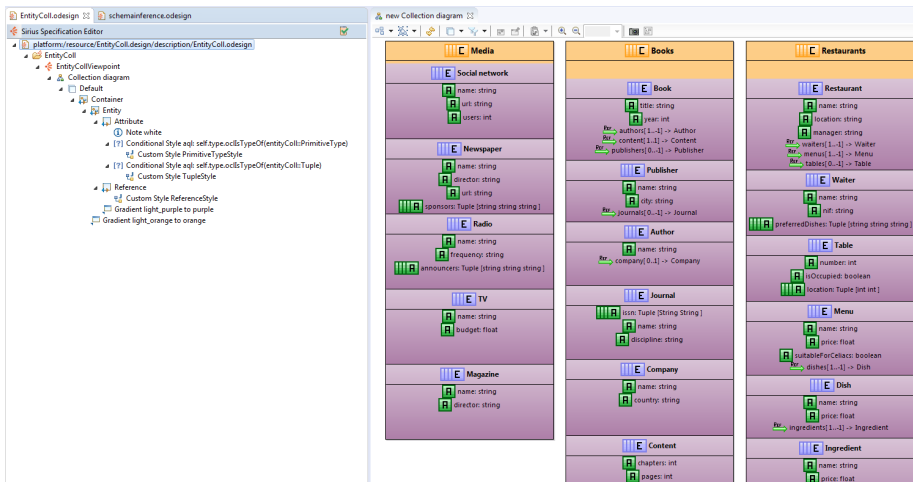


3. Horizontal stack layout

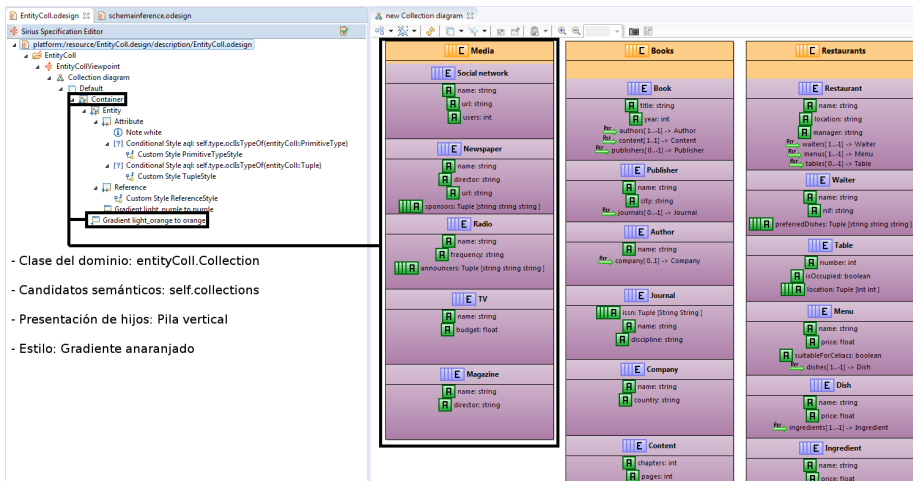


4. Vertical stack layout

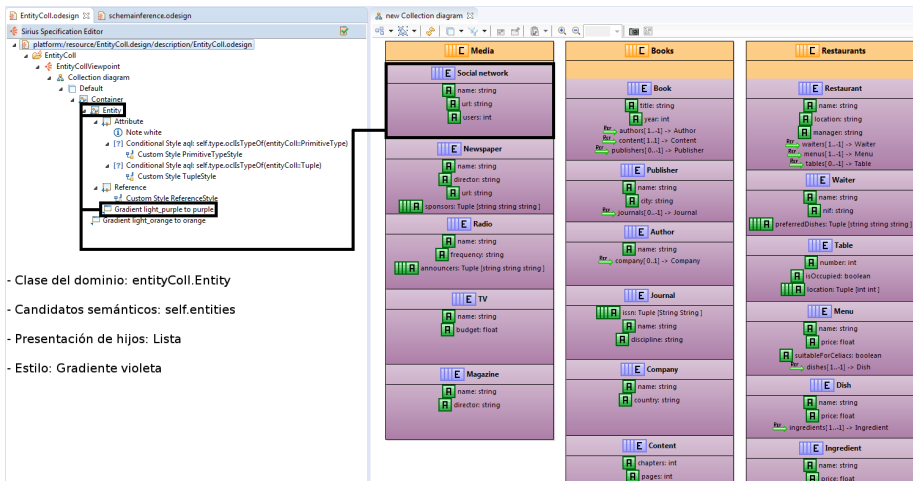
Aplicación de los conceptos básicos



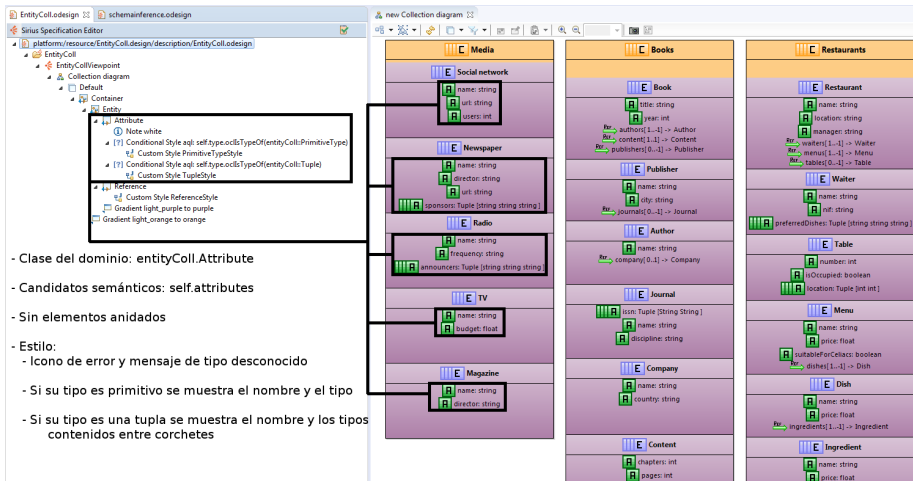
Aplicación de los conceptos básicos



Aplicación de los conceptos básicos

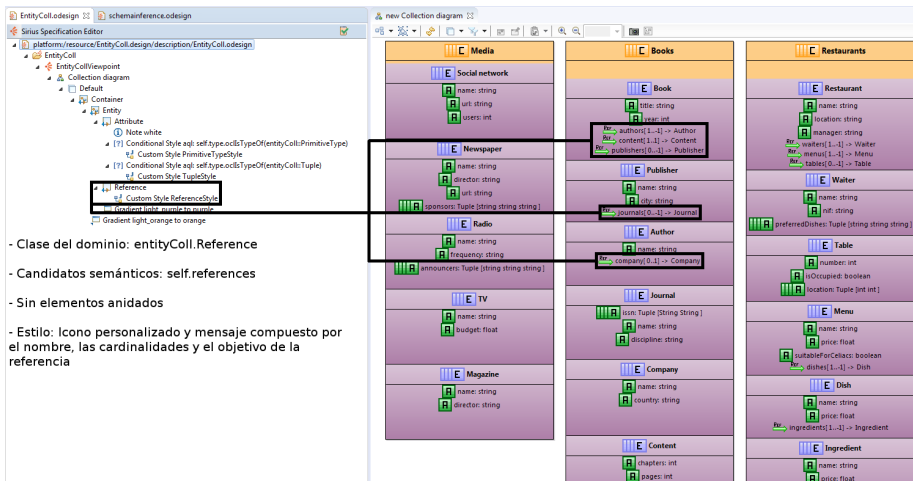


Aplicación de los conceptos básicos

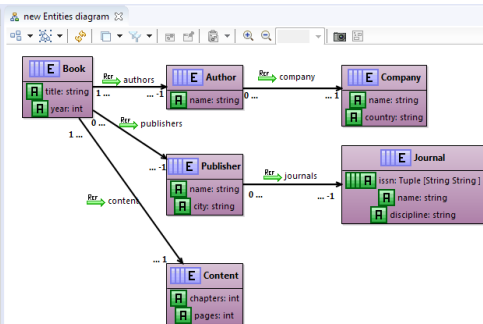
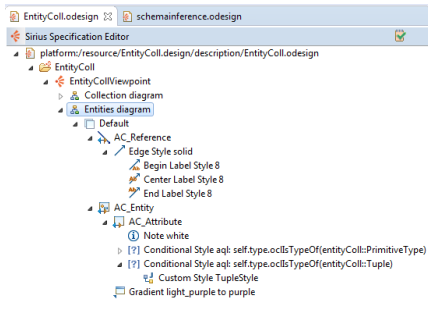


- Clase del dominio: entityColl.Attribute
- Candidatos semánticos: self.attributes
- Sin elementos anidados
- Estilo:
 - Icono de error y mensaje de tipo desconocido
 - Si su tipo es primitivo se muestra el nombre y el tipo
 - Si su tipo es una tupla se muestra el nombre y los tipos contenidos entre corchetes

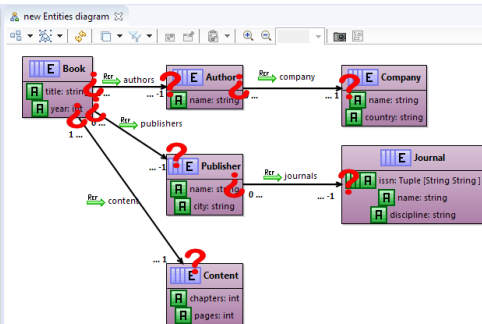
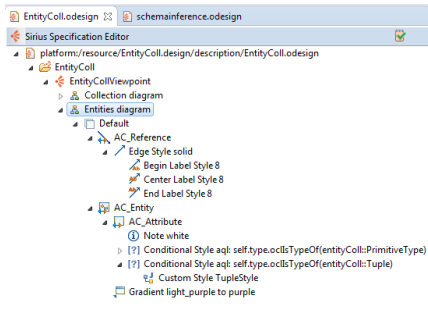
Aplicación de los conceptos básicos



Implementación de una vista para entidades

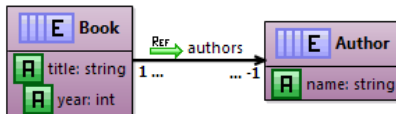


Implementación de una vista para entidades



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

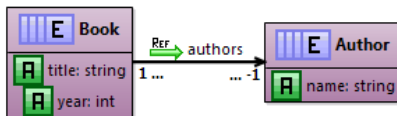


De nuevo es posible aplicar distintos estilos a los arcos:

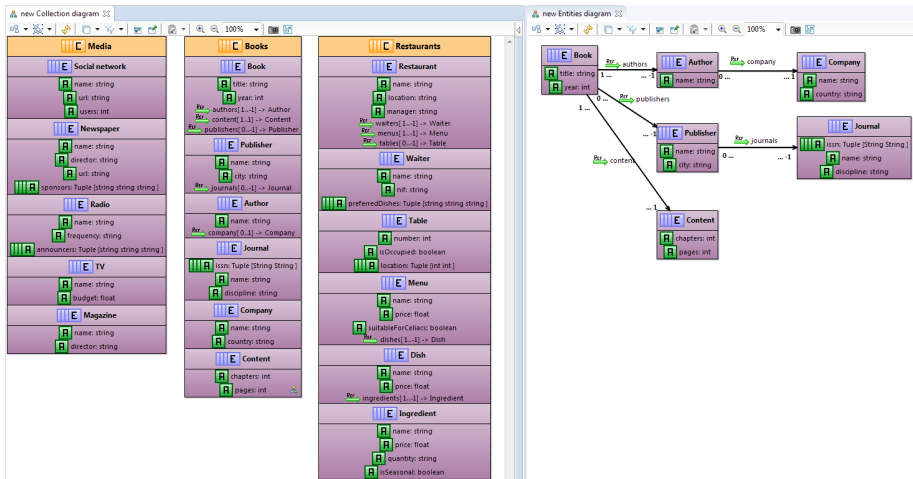
- Solidez y curvatura del trazo
- Decoradores al inicio y al final del arco
- Color, grosor, indicación de dónde debe terminar...

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



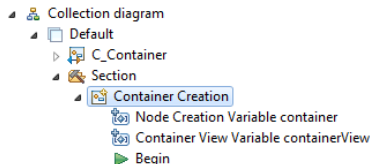
Vistas desarrolladas hasta ahora



Implementando la creación de elementos:

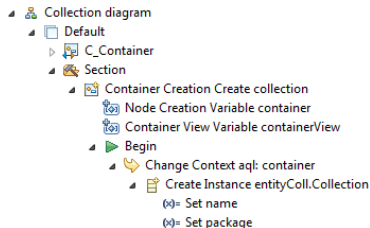
- Es posible agregar *secciones* a un diagrama
- En la sección se pueden agregar distinta funcionalidad:
 - Paleta de creación de elementos
 - Operaciones de interacción con elementos: Edición directa, reconexión de arcos, doble click, *drag & drop*...
 - Opciones de menú emergente
 - Navegación entre 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

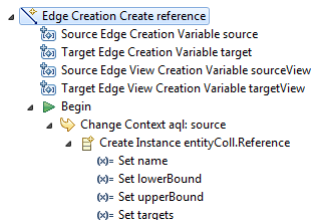
Reference name: collections, Type name: entityColl.Collections

- 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'. The main area shows three collection diagrams: 'Media', 'Books', and 'Restaurants'. Each diagram contains entities like 'Social network', 'Book', 'Restaurant', 'Newspaper', 'Publisher', 'Waiter', 'Radio', 'Table', 'TV', 'Magazine', 'Company', 'Content', 'Menu', 'Dish', and 'Ingredient', each with its attributes and relationships. The right pane shows the 'Palette' with icons for creating collections, entities, and attributes.

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 of the design structure. The 'Entities diagram' is selected, and the 'Section Creation palette' is visible, listing various creation actions such as 'Create entity', 'Create primitive attribute', 'Create tuple attribute', and 'Create reference'. The main workspace shows a 'new Entities diagram' with several entities and their relationships. The entities are represented by boxes with a blue header and a green body. The attributes are listed in the body. The relationships are represented by lines with labels and cardinalities. The entities and their attributes are: Book (title: string, year: int), Author (name: string), Company (name: string, country: string), Publisher (name: string, city: string), Journal (issn: Tuple [String String], name: string, discipline: string), and Content (chapters: int, pages: int). The relationships are: Book to Author (labeled 'Br' with 'authors'), Book to Publisher (labeled 'Br' with 'publishers'), Book to Content (labeled 'Br' with 'content'), Author to Company (labeled 'Br' with 'company'), Publisher to Journal (labeled 'Br' with 'journals'), and Journal to Content (labeled 'Br' with 'content'). The cardinalities are: Book to Author (1 to ...), Book to Publisher (1 to ...), Book to Content (1 to ...), Author to Company (0 to ...), Publisher to Journal (0 to ...), and Journal to Content (0 to ...).

EntityColl.odesign

Sirius Specification Editor

platform:/resource/EntityColl.design/description/EntityColl.odesign

- EntityColl
 - 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

new Entities diagram

Entities diagram

Entities and relationships:

- Book (title: string, year: int)
- Author (name: string)
- Company (name: string, country: string)
- Publisher (name: string, city: string)
- Journal (issn: Tuple [String String], name: string, discipline: string)
- Content (chapters: int, pages: int)

Relationships:

- Book to Author (Br, authors, 1 to ...)
- Book to Publisher (Br, publishers, 1 to ...)
- Book to Content (Br, content, 1 to ...)
- Author to Company (Br, company, 0 to ...)
- Publisher to Journal (Br, journals, 0 to ...)
- Journal to Content (Br, content, 0 to ...)

Palette

Creation palette

- 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

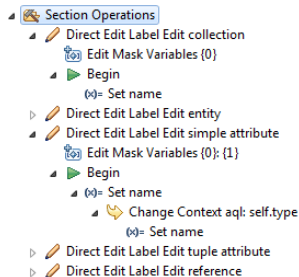
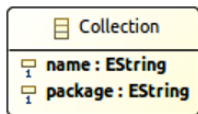
El editor puede enriquecerse con las siguientes utilidades:

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

Tenemos un DSL gráfico...¡sin interacción!

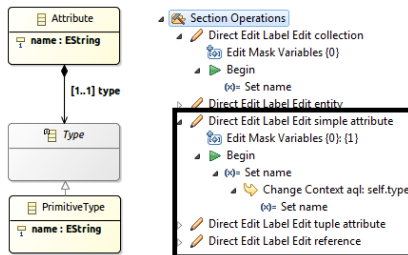
- Sirius permite crear secciones con interacciones:
 - Eliminar elementos
 - Reconectar arcos
 - Doble click sobre elementos
 - Edición de mensajes
 - *Drag & drop*
- Las interacciones se implementan haciendo uso de las operaciones genéricas:
 - *Cambio de contexto, crear instancia, if, set, unset*
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

New Element Edition \Rightarrow Direct edit label



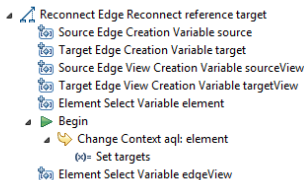
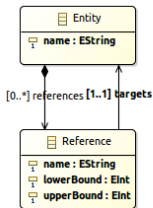
- Se debe indicar un identificador y la metaclase asociada
- Aplicar una máscara: {0} por defecto
- Set name \Rightarrow var:0

New Element Edition \Rightarrow Direct edit label



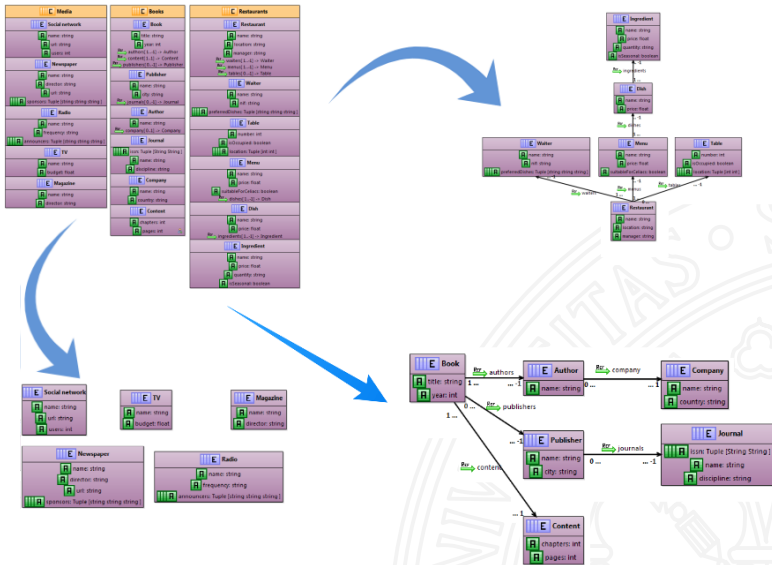
- Máscara {0}: {1}
- Set name \Rightarrow var:0
- *Cambio de contexto*: Para poder modificar el atributo *name* del tipo
aql: self.type
- Set name \Rightarrow var:1

New Element Edition \Rightarrow Reconnect edge

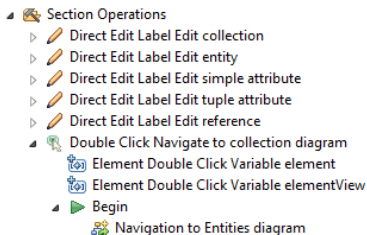


- Es posible reconectar en origen y/o objetivo de un arco
- Para reducir la complejidad se recomienda crear dos operaciones
- Variables importantes: *source*, *target*, *element*
 - *Cambio de contexto*: Para movernos a la clase *Reference*
aql: element
 - Set targets \Rightarrow aql: target

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 metacalse
- 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 'Filter attributes' filter, listing the following options:

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

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

- Books**: Book (name: string, price: int, author: string, publisher: string, year: int). Filters: authors(1..*) -> Author, content(1..*) -> Content, publisher(0..*) -> Publisher.
- Restaurants**: Restaurant (name: string, location: string, manager: string, waiter: string, menu: string, table: string). Filters: waiter(1..*) -> Waiter, menu(1..*) -> Menu, table(0..*) -> Table.
- Publisher**: Publisher (name: string, city: string, journal: string). Filter: journal(0..*) -> Journal.
- Author**: Author (name: string, company: string). Filter: company(0..*) -> Company.
- Journal**: Journal (name: string, discipline: string, year: string). Filter: year(0..*) -> Year.
- Company**: Company (name: string, country: string). Filter: country(0..*) -> Country.
- Content**: Content (chapters: int, pages: int).
- TV**: TV (name: string, budget: float, announcers: Tuple(string string string)).
- Radio**: Radio (name: string, frequency: string).
- Magazine**: Magazine (name: string, director: string).
- Table**: Table (number: int, occupied: boolean, location: Tuple(int int)).
- Menu**: Menu (name: string, price: float, suitableForCafe: boolean). Filter: suitableForCafe(1..*) -> Dish.
- Dish**: Dish (name: string, price: float, ingredients: string). Filter: ingredients(1..*) -> Ingredient.
- Ingredient**: Ingredient (name: string, price: float, quantity: string, isSeasonal: boolean).

Se pueden definir reglas de validación en Sirius:

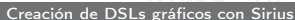
- Para asegurar la corrección del modelo formado
- Con capacidad para sugerir cambios, arreglos y gravedad del error
- **New validation** \Rightarrow **Validation** \Rightarrow **Semantic validation rule**
- Para cada regla de validación se debe indicar:
 - La importancia de la regla: *Information*, *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())`

Alberto Hernández, Jesús García Molina



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**: Contains the following Java 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:
 - 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 two elements:
 - Collection_5** (orange box)
 - Brand_New_Entity_Name** (purple box)

At the bottom, the **Properties** panel is open, showing the **Set name** property:

- General** tab is selected.
- Feature Name:** ? name
- Value Expression:** ? service: getEntityTitle

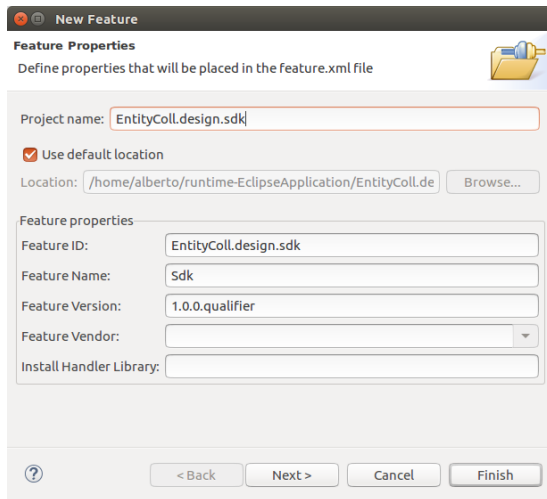
Índice de contenido

- 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:

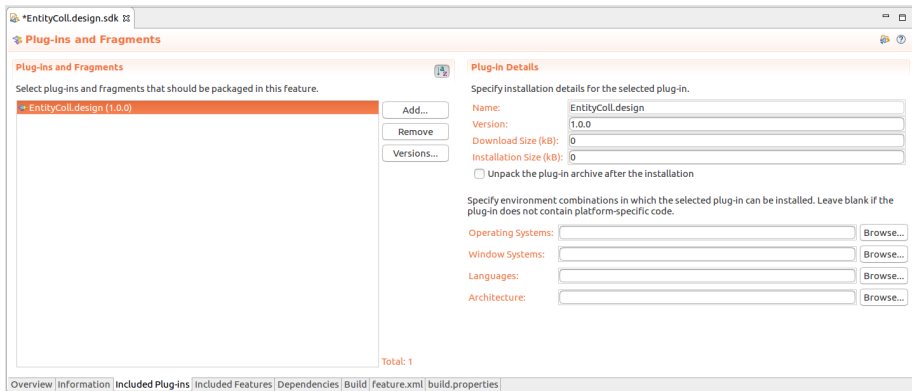
Feature Version:

Feature Vendor: 

Install Handler Library:

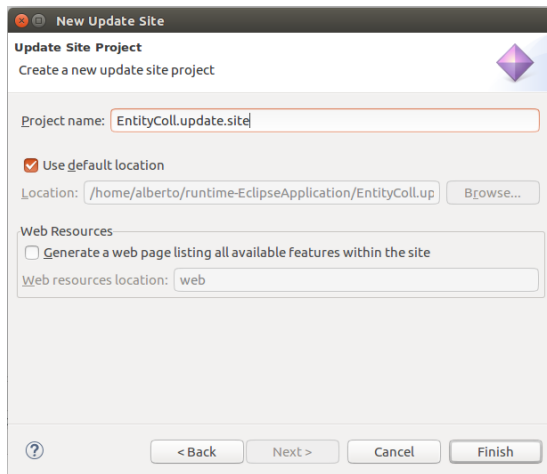


Inclusión del proyecto *design*



Creación de un *Update site*

File ⇒ New ⇒ Update site



The screenshot shows the 'New Update Site' dialog box. The title bar reads 'New Update Site'. Below the title bar, the text 'Update Site Project' is followed by 'Create a new update site project'. A purple diamond icon is on the right. The 'Project name:' field contains 'EntityColl.update.site'. The 'Use default location' checkbox is checked. The 'Location:' field contains '/home/alberto/runtime-EclipseApplication/EntityColl.up', with a 'Browse...' button to its right. The 'Web Resources' section has an unchecked checkbox for 'Generate a web page listing all available features within the site' and a 'Web resources location:' field containing 'web'. At the bottom, there is a help icon (?), and buttons for '< Back', 'Next >', 'Cancel', and 'Finish'.

New Update Site

Update Site Project
Create a new update site project

Project name: EntityColl.update.site

☒ Use default location

Location: /home/alberto/runtime-EclipseApplication/EntityColl.up Browse...

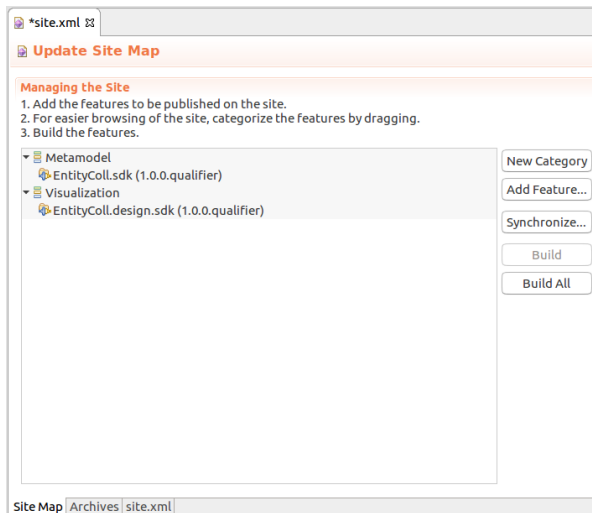
Web Resources

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

Web resources location: web

? < Back Next > Cancel Finish

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>