

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



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

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](#)

2016201520142013

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

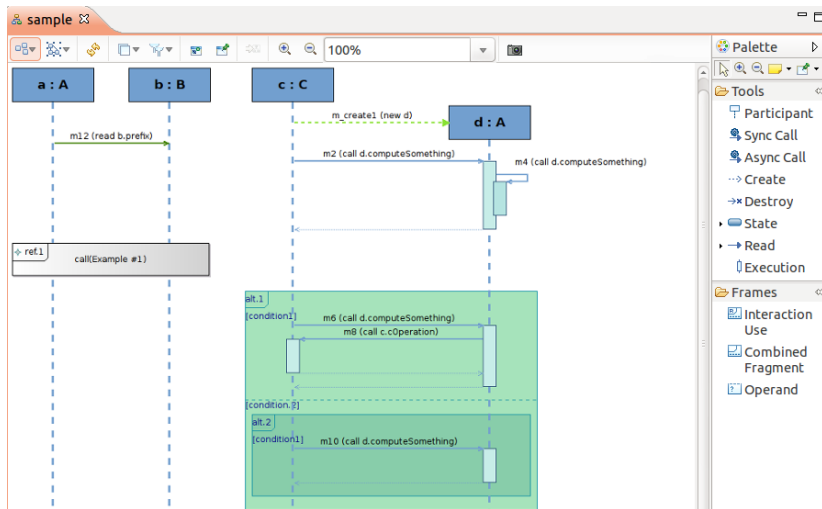
(<http://eclipse.org/sirius/>)

- 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!

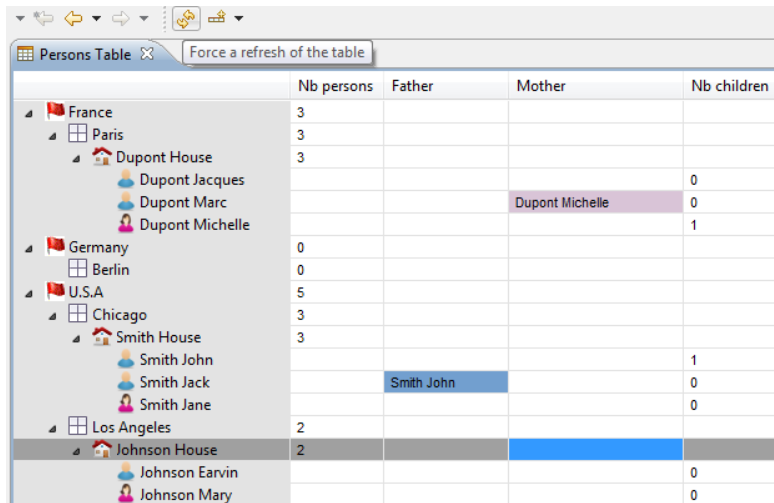
Sirius se basa en los siguientes puntos:

- 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



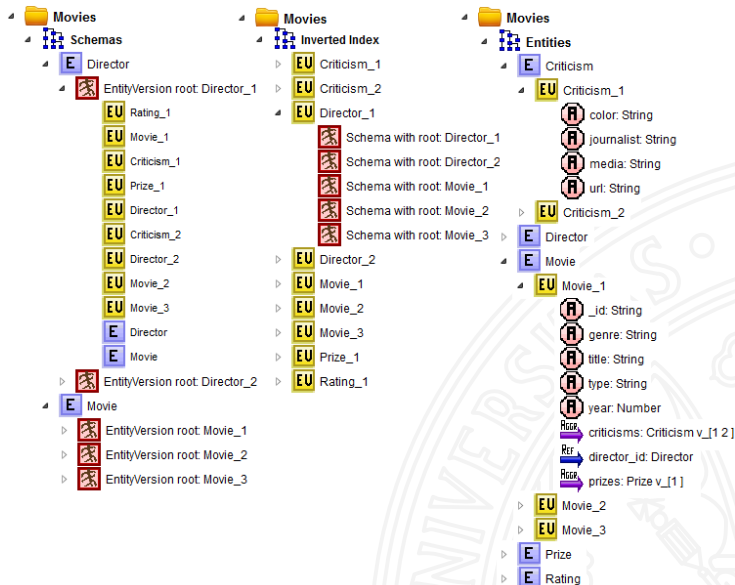
DSLs gráficos - Tablas



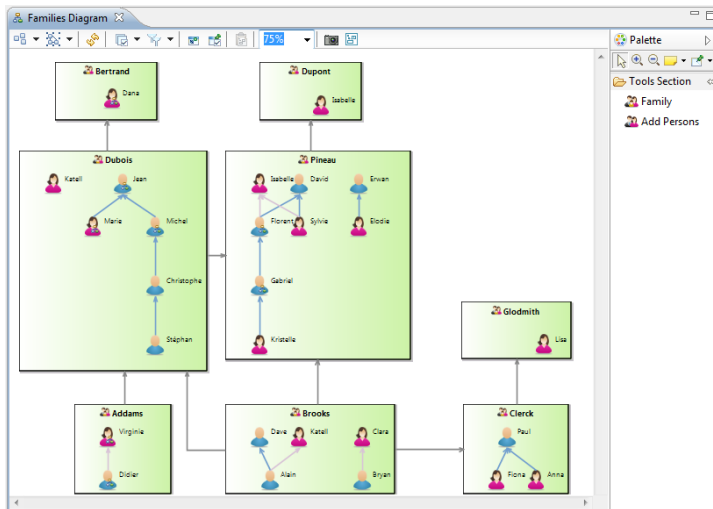
Persons Table Force a refresh of the table

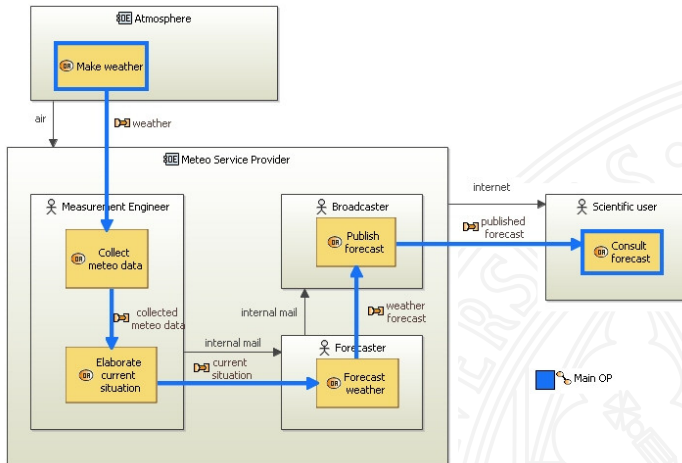
	Nb persons	Father	Mother	Nb children
France	3			
Paris	3			
Dupont House	3			
Dupont Jacques				0
Dupont Marc			Dupont Michelle	0
Dupont Michelle				1
Germany	0			
Berlin	0			
U.S.A	5			
Chicago	3			
Smith House	3			
Smith John				1
Smith Jack		Smith John		0
Smith Jane				0
Los Angeles	2			
Johnson House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

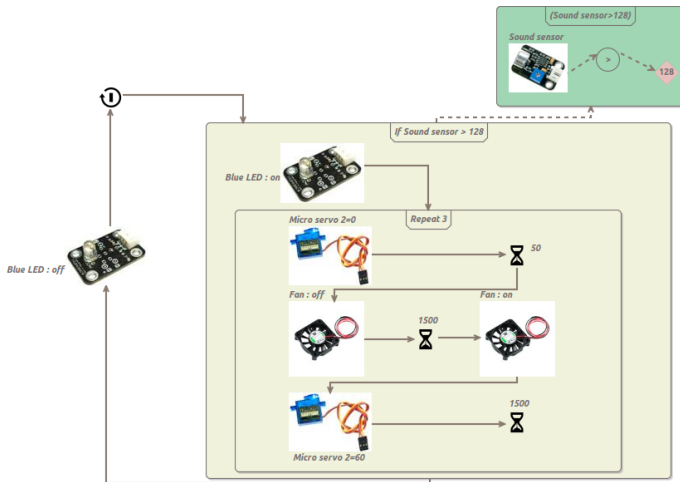
DSLs gráficos - Árboles



DSLs gráficos - Diagramas



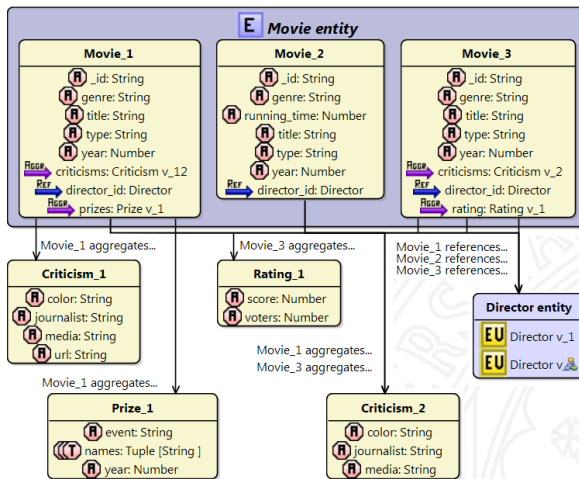




DSLs gráficos - Galería (III)




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](#) ➔

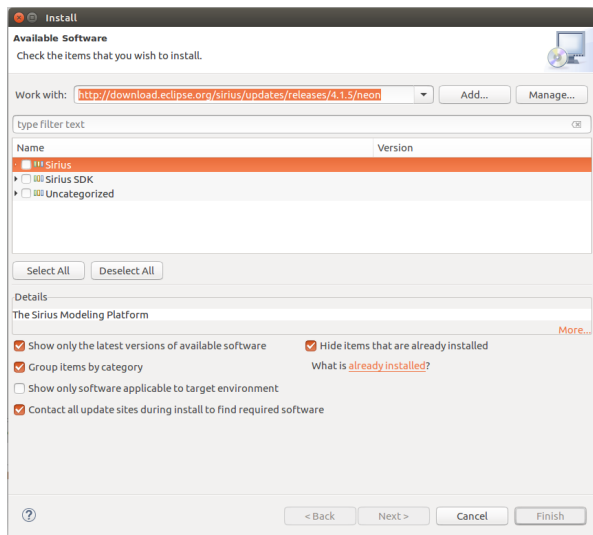
Need collaborative features for Sirius?

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>)

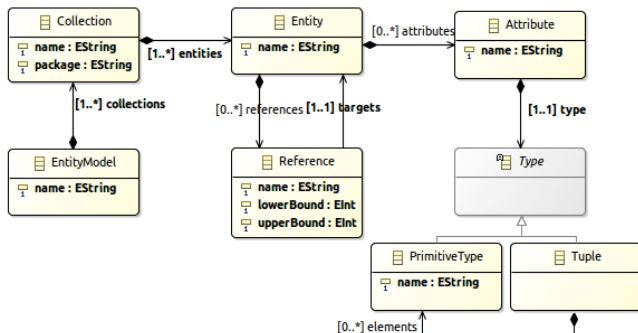
Instalación de Sirius y componentes (II)



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



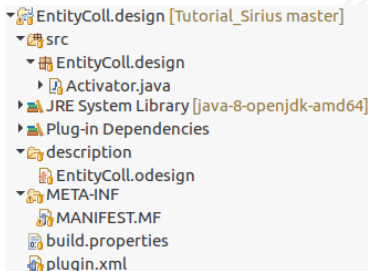
- ✦ Entity Model Stuff
 - ✦ Collection Media
 - ✦ Entity Social network
 - ✦ Entity Newspaper
 - ✦ Entity Radio
 - ✦ Entity TV
 - ✦ Entity Magazine
 - ✦ Collection Books
 - ✦ Entity Book
 - ✦ Entity Publisher
 - ✦ Entity Author
 - ✦ Entity Journal
 - ✦ Entity Company
 - ✦ Entity Content
 - ✦ Collection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

Escenario de partida:

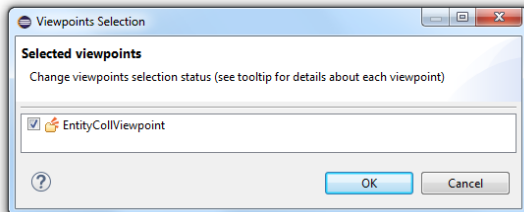
- Se dispone de un metamodelo instalado en Eclipse
- Se dispone de la última versión de Sirius

File ⇒ New ⇒ Viewpoint Specification Project:

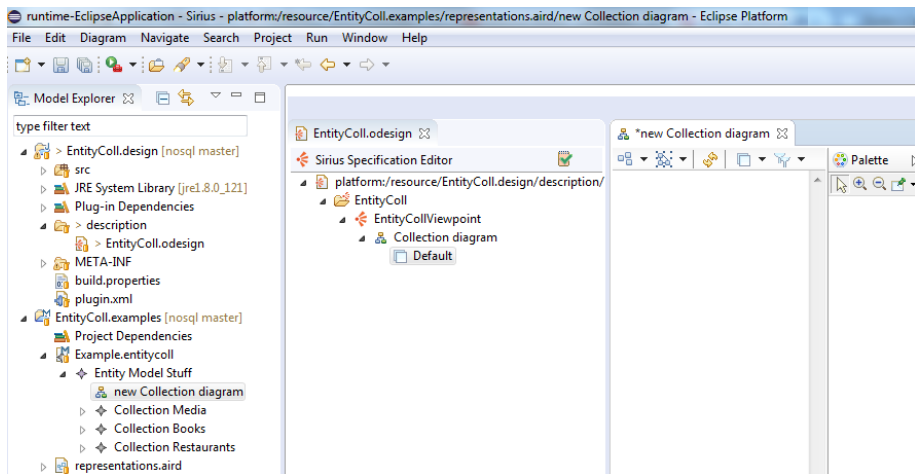
- Implementado a partir de un *Eclipse plug-in project*



- 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 for creating visual elements. The top-left pane displays the project structure:

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

The top-right pane shows a toolbar with various icons for creating and editing diagrams. The bottom-left pane shows the 'Properties' view for the 'Collection' element. The 'General' tab is selected, and the 'Children Presentation*' section is highlighted. The 'Id*' field is set to 'Collection', and the 'Label' is also 'Collection'. The 'Domain Class*' is set to 'entityColl.Collection'. The 'Semantic Candidates Expression' is set to 'aqi: self.collections'. The 'Children Presentation*' section has four radio buttons: 'Free Form' (selected), 'List', 'Horizontal Stack', and 'Vertical Stack'.

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 'General' tab is selected. The 'Children Presentation*' section is highlighted with a red question mark.

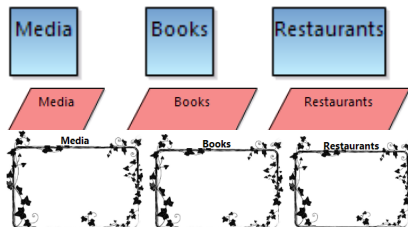
Properties View:

- 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

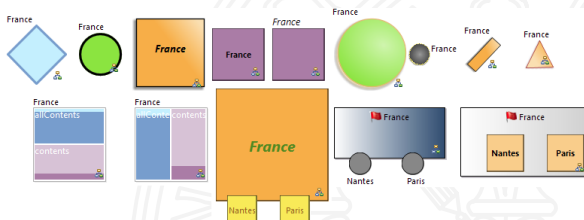
A cada contenedor se le pueden aplicar los siguientes estilos:

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



A cada nodo se le pueden aplicar los siguientes estilos:

- *Workspace image*
- *Formas geométricas*

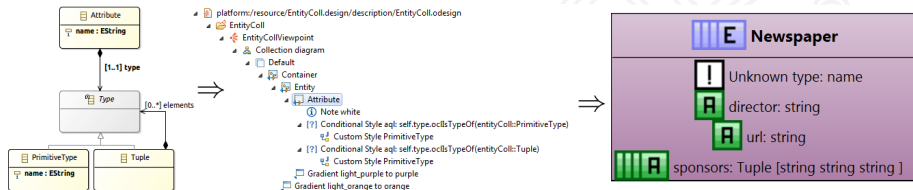


Es posible modificar muchos aspectos de los estilos:

- 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...



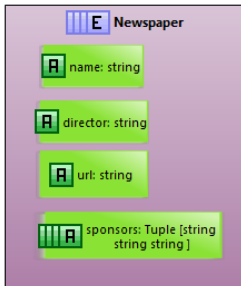
Creando entidades visuales en Sirius

The screenshot displays the Sirius IDE interface. The top-left pane shows the 'Sirius Specification Editor' with a project tree containing 'platform:/resource/EntityColl.design/description/EntityColl.odesign', 'EntityColl', 'EntityCollViewpoint', 'Collection diagram', 'Default', and '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:

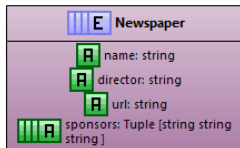
- Id*:** Collection
- Label:** Collection
- Domain Class*:** entityColl.Collection
- Semantic Candidates Expression:** aql: self.collections
- Children Presentation*:** Free Form (selected), List, Horizontal Stack, Vertical Stack

Red question marks are placed over the 'Children Presentation*' label and the 'Free Form' radio button.

Tipos de layouts para contenedores



1. Free form layout



2. List layout

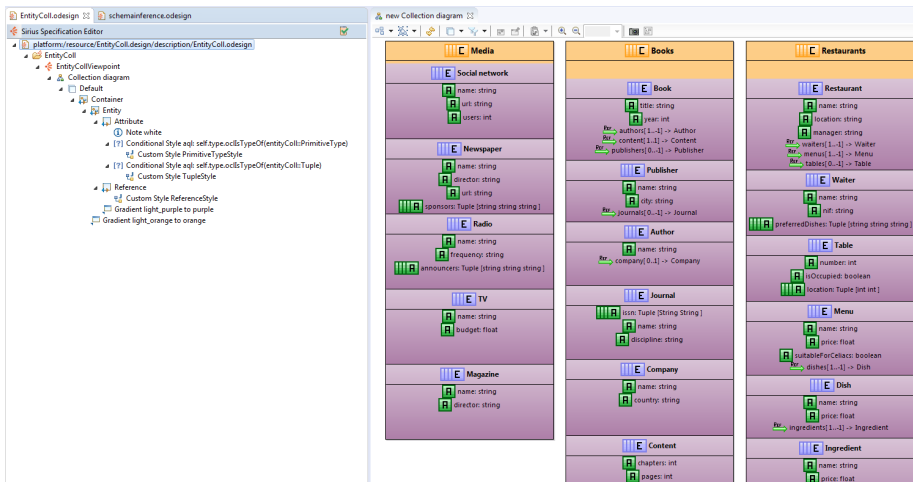


3. Horizontal stack layout



4. Vertical stack layout

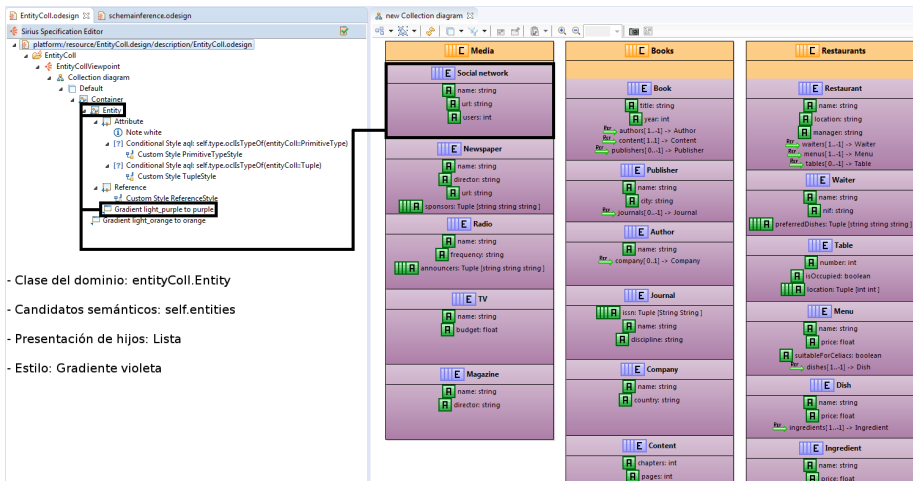
Aplicación de los conceptos básicos



Aplicación de los conceptos básicos

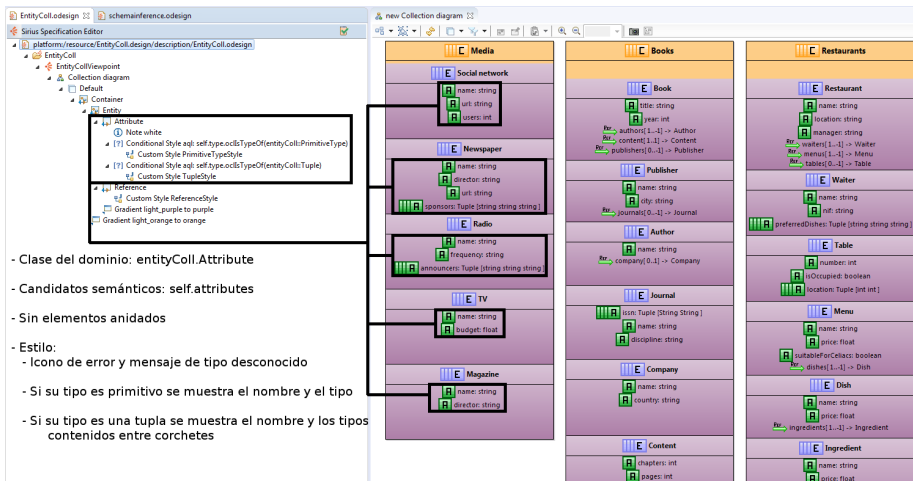


Aplicación de los conceptos básicos



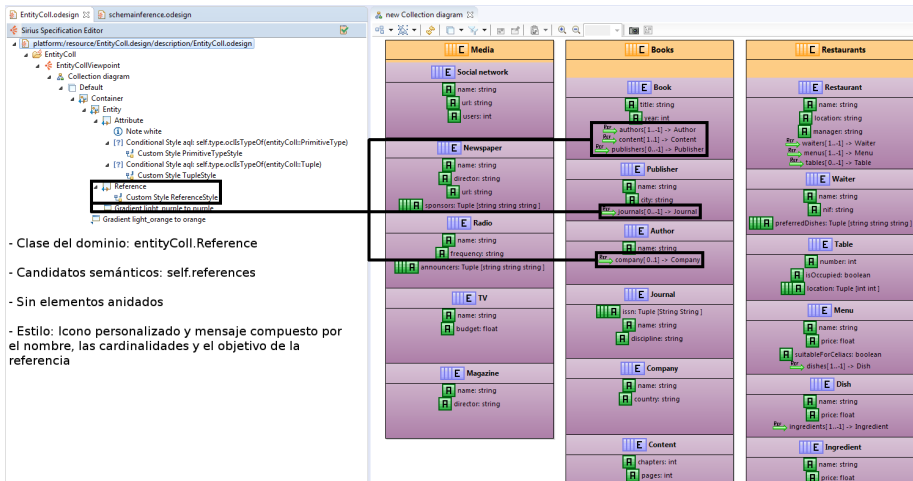
- Clase del dominio: entityColl.Entity
- Candidatos semánticos: self.entities
- Presentación de hijos: Lista
- Estilo: Gradiente violeta

Aplicación de los conceptos básicos



- 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



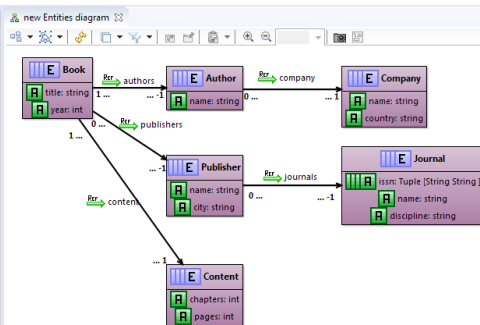
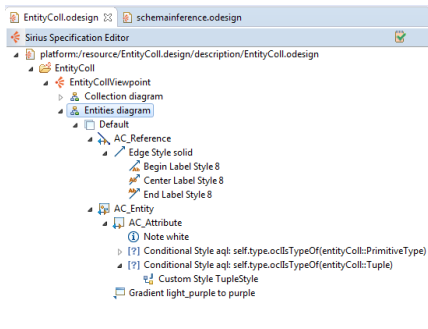
- Clase del dominio: entityColl.Reference

- Candidatos semánticos: self.references

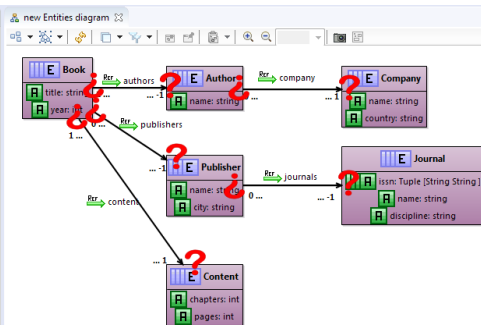
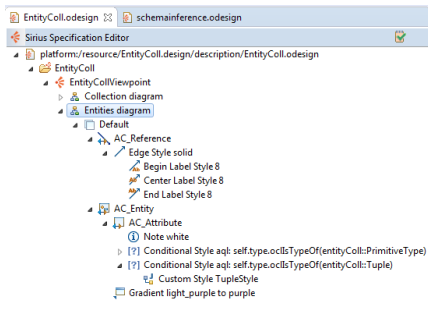
- Sin elementos anidados

- Estilo: Icono personalizado y mensaje compuesto por el nombre, las cardinalidades y el objetivo de la referencia

Implementación de una vista para entidades

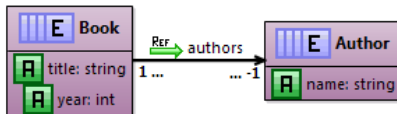


Implementación de una vista para entidades



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

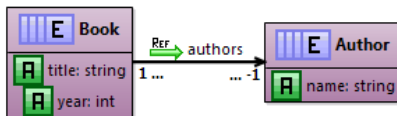


De nuevo es posible aplicar distintos estilos a los arcos:

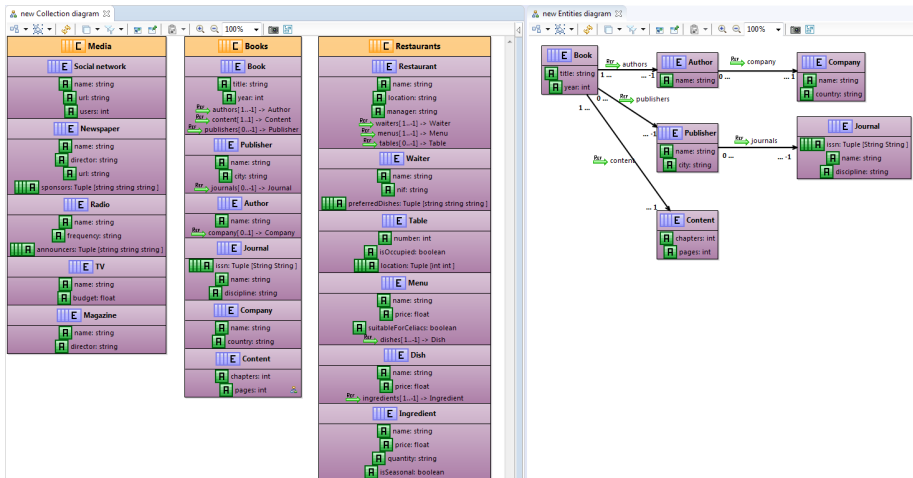
- Solidez y curvatura del trazo
- Decoradores al inicio y al final del arco
- Color, grosor, indicación de dónde debe terminar...

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



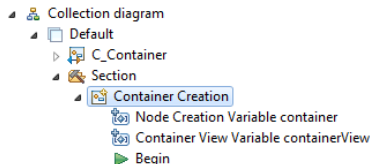
Vistas desarrolladas hasta ahora



Implementando la creación de elementos:

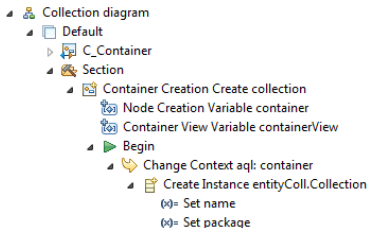
- Es posible agregar *secciones* a un diagrama
- En la sección se pueden agregar distinta funcionalidad:
 - Paleta de creación de elementos
 - Operaciones de interacción con elementos: Edición directa, reconexión de arcos, doble click, *drag & drop*...
 - Opciones de menú emergente
 - Navegación entre 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 displays the Sirius Specification Editor interface for a project named 'EntityColl.odesign'. The left-hand pane shows the project structure, including 'platform/resource/EntityColl.design/description/EntityColl.odesign', 'EntityColl', 'EntityCollectionViewpoint', 'Collection diagram', 'Default', 'C.Container', 'Section', 'Container Creation Create collection', 'Container Creation Create entity', 'Node Creation Create primitive attribute', 'Node Creation Variable container', 'Container View Variable containerView', 'Begin', 'Change Context aql: container', 'Create Instance entityColl.PrimitiveType', 'Set name', 'Node Creation Create tuple attribute', 'Node Creation Create reference', 'Node Creation Variable container', 'Container View Variable containerView', 'Begin', 'Change Context aql: container', 'Create Instance entityColl.Reference', 'Set name', 'Set lowerBound', and 'Set upperBound'. The main area shows three collection diagrams: 'Media', 'Books', and 'Restaurants'. The 'Media' diagram includes entities like 'Social network', 'Newspaper', 'Radio', 'TV', and 'Magazine'. The 'Books' diagram includes entities like 'Book', 'Publisher', 'Author', 'Company', and 'Journal'. The 'Restaurants' diagram includes entities like 'Restaurant', 'Waiter', 'Table', 'Menu', 'Dish', and 'Ingredient'. The right-hand pane shows the 'Palette' with creation tools for collections, entities, and attributes.

new Collection diagram

Media

- E Social network**
 - A name**: string
 - A url**: string
 - A users**: int
- E Newspaper**
 - A name**: string
 - A director**: string
 - A url**: string
- A sponsors**: Tuple (string string string)
- E Radio**
 - A name**: string
 - A frequency**: string
- A announcers**: Tuple (string string string)
- E TV**
 - A name**: string
 - A budget**: float
- E Magazine**
 - A name**: string
 - A director**: string

Books

- E Book**
 - A title**: string
 - A year**: int
- author(s) 1..2** → Author
- company 1..2** → Company
- publisher(s) 0..2** → Publisher
- E Publisher**
 - A name**: string
 - A city**: string
- journal(s) 0..2** → Journal
- E Author**
 - A name**: string
 - A company**(0..2) → Company
- E Journal**
 - A issn**: Tuple (string string)
 - A name**: string
 - A discipline**: string
- E Company**
 - A name**: string
 - A country**: string
- E Content**
 - A chapters**: int
 - A pages**: int

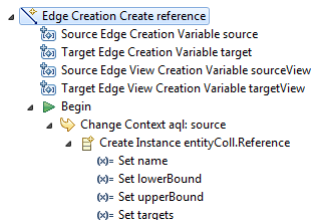
Restaurants

- E Restaurant**
 - A name**: string
 - A location**: string
 - A manager**: string
- waiter(s) 1..2** → Waiter
- menu(s) 1..2** → Menu
- table(s) 0..2** → Table
- E Waiter**
 - A name**: string
 - A nfr**: string
- A preferredDishes**: Tuple (string string string)
- E Table**
 - A number**: int
 - A occupied**: boolean
- A location**: Tuple (int int)
- E Menu**
 - A name**: string
 - A price**: float
- A availableOnCalendar**: boolean
- dish(es) 1..1** → Dish
- E Dish**
 - A name**: string
 - A price**: float
- ingredient(s) 1..2** → Ingredient
- E Ingredient**
 - A name**: string
 - A price**: float
 - A quantity**: string
 - A seasonal**: boolean

Palette

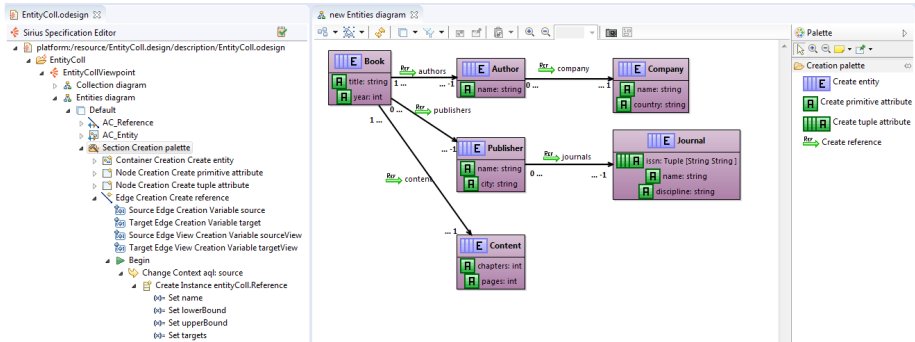
- Create collection
- Create entity
- Create primitive attribute
- Create tuple attribute
- Create reference

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



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

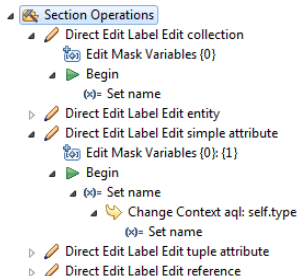
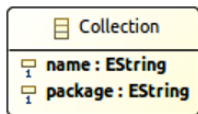
El editor puede enriquecerse con las siguientes utilidades:

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

Tenemos un DSL gráfico...¡sin interacción!

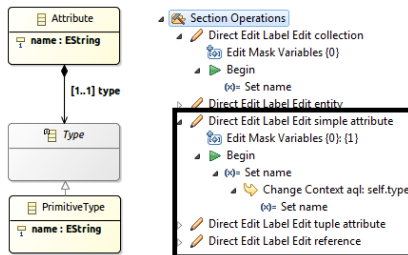
- Sirius permite crear secciones con interacciones:
 - Eliminar elementos
 - Reconectar arcos
 - Doble click sobre elementos
 - Edición de mensajes
 - *Drag & drop*
- Las interacciones se implementan haciendo uso de las operaciones genéricas:
 - *Cambio de contexto, crear instancia, if, set, unset*
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

New Element Edition \Rightarrow Direct edit label



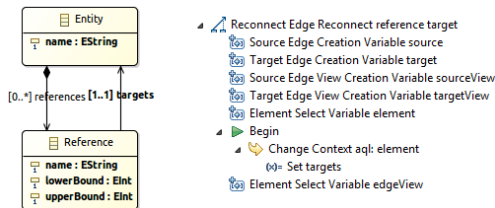
- Se debe indicar un identificador y la metaclase asociada
- Aplicar una máscara: {0} por defecto
- Set name \Rightarrow var:0

New Element Edition \Rightarrow Direct edit label



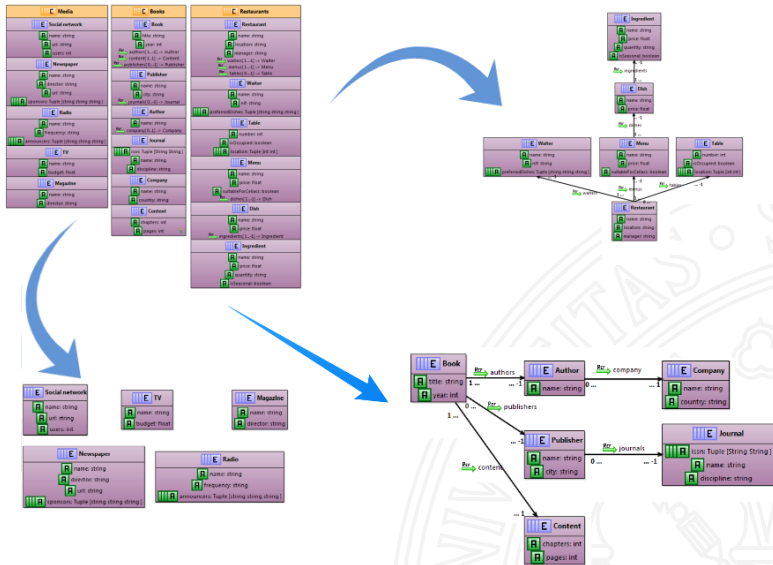
- Máscara {0}: {1}
- Set name \Rightarrow var:0
- *Cambio de contexto*: Para poder modificar el atributo *name* del tipo
aql: self.type
- Set name \Rightarrow var:1

New Element Edition \Rightarrow Reconnect edge

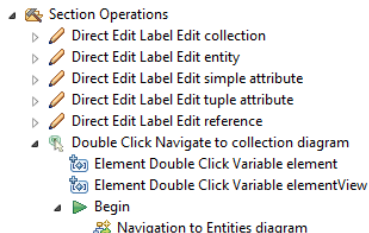


- Es posible reconectar en origen y/o objetivo de un arco
- Para reducir la complejidad se recomienda crear dos operaciones
- Variables importantes: *source*, *target*, *element*
 - *Cambio de contexto*: Para movernos a la clase *Reference*
aql: element
 - Set targets \Rightarrow aql: target

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, and its configuration is shown in the main editor. The configuration includes a list of filters: 'Filter attributes', 'Filter entities', 'Filter references', and 'Filter empty objects'. The main editor shows a collection diagram with several entities: 'Books', 'Newspaper', 'Radio', 'TV', 'Publisher', 'Author', 'Journal', 'Company', 'Content', 'Restaurants', 'Restaurant', 'Table', 'Menu', 'Dish', and 'Ingredient'. Each entity is represented by a box with a filter icon (a green square with a white 'A') and a list of attributes. The 'Filter attributes' filter is applied to the 'Books' entity, and the 'Filter entities' filter is applied to the 'Newspaper' entity. The 'Filter references' filter is applied to the 'Publisher' entity, and the 'Filter empty objects' filter is applied to the 'Radio' entity. The 'Filter attributes' filter is also applied to the 'TV' entity. The 'Filter entities' filter is applied to the 'Publisher' entity. The 'Filter references' filter is applied to the 'Author' entity. The 'Filter empty objects' filter is applied to the 'Journal' entity. The 'Filter attributes' filter is applied to the 'Company' entity. The 'Filter entities' filter is applied to the 'Content' entity. The 'Filter references' filter is applied to the 'Restaurants' entity. The 'Filter empty objects' filter is applied to the 'Restaurant' entity. The 'Filter attributes' filter is applied to the 'Table' entity. The 'Filter entities' filter is applied to the 'Menu' entity. The 'Filter references' filter is applied to the 'Dish' entity. The 'Filter empty objects' filter is applied to the 'Ingredient' entity.

EntityColl.odesign

Sirius Specification Editor

platforms/resource/EntityColl.design/description/Ent

EntityColl

EntityCollViewpoint

Collection diagram

Filter attributes

Mapping Filter HIDE

Filter entities

Mapping Filter HIDE

Filter references

Mapping Filter HIDE

Filter empty objects

Mapping Filter HIDE

Mapping Filter HIDE

Default

C.Container

Section Creation palette

Section Operations

Entities diagram

new Collection diagram

75%

Filter attributes

Filter entities

Filter references

Filter empty objects

Books

Book

name: string

price: int

author: 1..* -> Author

content: 1..* -> Content

publisher: 0..1 -> Publisher

Newspaper

name: string

director: string

url: string

sponsor: Tuple (string string)

Radio

name: string

frequency: string

announcers: Tuple (string string)

TV

name: string

budget: float

Magazine

name: string

director: string

Publisher

name: string

city: string

Journal

name: string

company: 0..1 -> Company

Journal

name: string

discipline: string

Company

name: string

country: string

Content

chapters: int

pages: int

Restaurants

Restaurant

name: string

location: string

manager: string

waiters: 1..* -> Waiter

menu: 1..* -> Menu

tables: 0..* -> Table

Waiter

name: string

shift: string

preferredDishes: Tuple (string string)

Table

number: int

isOccupied: boolean

location: Tuple (int int)

Menu

name: string

price: float

suitableForCeliac: boolean

dishes: 1..* -> Dish

Dish

name: string

price: float

ingredient: 1..* -> Ingredient

Ingredient

name: string

price: float

quantity: string

isSeasonal: boolean

Se pueden definir reglas de validación en Sirius:

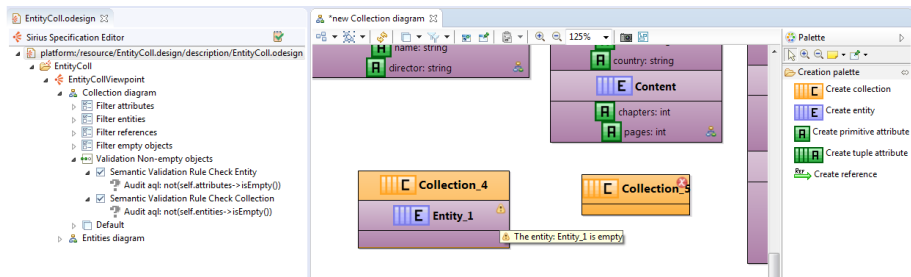
- Para asegurar la corrección del modelo formado
- Con capacidad para sugerir cambios, arreglos y gravedad del error
- **New validation** \Rightarrow **Validation** \Rightarrow **Semantic validation rule**
- Para cada regla de validación se debe indicar:
 - La importancia de la regla: *Information*,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

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**: 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 active, showing the **Set name** property:

- General** tab
- Feature Name:**
- Value Expression:**

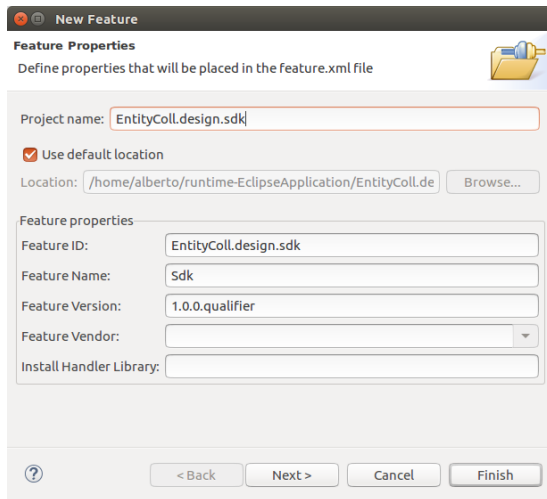
Índice de contenido

- 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:

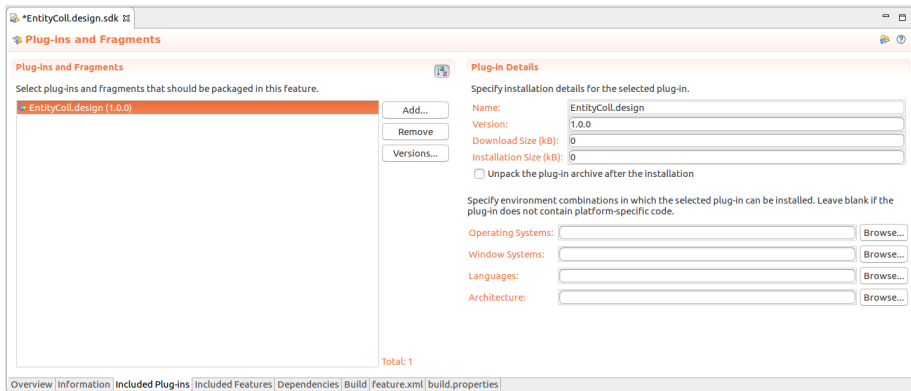
Feature Name:

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

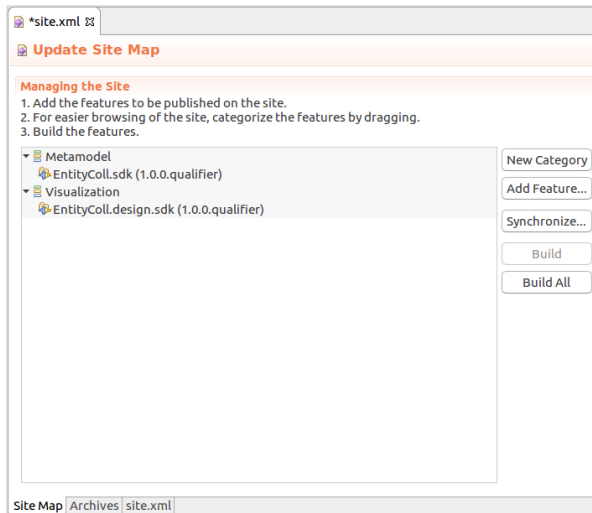
Location:

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>