Herramienta CASE: Java desde UML con XML y CLIPS.

Carlos Fernández de la Torre

unir LA UNIVERSIDAD EN INTERNET

ÍNDICE

unir LA UNIVERSIDAD EN INTERNET

01 – Introducción

Objetivo del Trabajo:

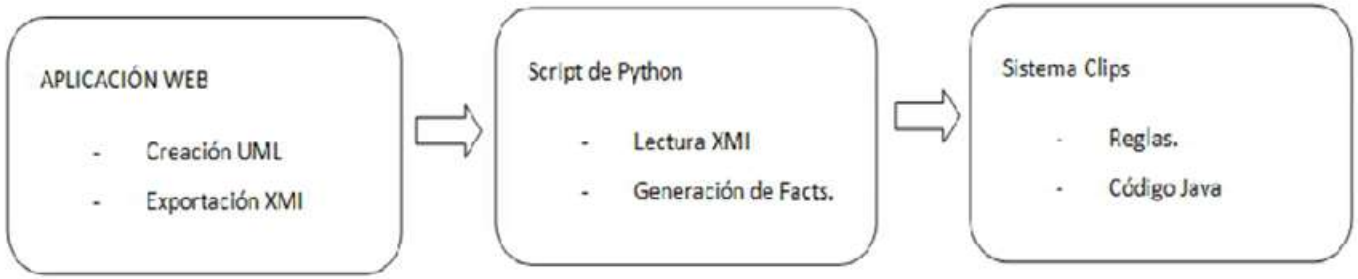
Desarrollo de una herramienta **CASE** de software **libre**.
Generación automática de código **Java** a partir de diagramas de clases **UML**.

Antecedentes:

Primeros intentos de automatización en los **años 70** con ISDOS (PSL y PSA) *(Mª Luisa Garzón V., 2007)*.
Uso de **patrones de diseño** como soluciones probadas para problemas específicos *(Pressman, 1993)*.

Proceso de Generación de Código:

Creación de diagramas UML (mejora el análisis y diseño OO de los estudiantes) *(Parga, 2015)*.
Conversión de diagramas **UML** en **formato gráfico a texto** (XML, SVG, XMI) *(Timothy J. Grose, 2003)*.
Parser que convierte XMI en **hechos y reglas** de CLIPS *(Soriano Zárate, 2004)*.



01 – Introducción
Justificación

Relevancia de la POO y UML:

La **POO** es ampliamente adoptada por su capacidad para **modelar sistemas complejos** *(Martín, 2019).*
Ventajas de la POO sobre la Programación Estructurada son el **diseño y mantenimiento** *(Vanegas, 2001).*
UML **mejora la comunicación** y reduce la complejidad de la representación del sistema *(Booch, 2005).*

Ventajas de las Herramientas CASE:

Aumentan la **productividad y la calidad del código generado** *(Loucopoulos, 1995).*
El código es **más seguro y fácil de mantener** que el desarrollado manualmente *(Picón, 2016).*
Uso de **ingeniería inversa en el análisis de código,** generando modelos más abstractos *(Fontela, 2012),* además cuentan con **críticas de diseño** sugiriendo posibles mejoras *(Alicia Ramos Martín, 2014).*

Problema Central:

Falta de herramientas adecuadas (licencia, código generado ...) para **entornos educativos**.

Motivación del Proyecto:

Desarrollo una **herramienta libre** que facilite **el aprendizaje práctico de la POO**.
UML mejora de la **capacidad de abstracción** y **análisis formal** *(Serna M., 2011),* habilidades clave en la formación de programadores.
Que el alumnado vea como se **integran distintas tecnologías** en la elaboración de un proyecto.

02 – Contexto y Estado del Arte.

Generación Automática de Código y Modelos Avanzados:

La **IA** ha revolucionado la **generación de código**, con el uso de Grandes Modelos de Lenguaje (**LLM**), basados en arquitecturas de **Transformers** *(Shin, 2021; Mialon, 2023).*
Herramientas como GitHub Copilot y OpenAI Codex generan código desde **lenguaje natural,** gracias a la capacidad de los LLM para analizar y **generar tokens** del código fuente *(Abutridy, 2023).*

Ventajas e Inconvenientes de los LLM en la Generación de Código:

Ventajas: Automatización de tareas repetitivas, mejora de **productividad**, asistencia en lenguajes desconocidos, y generación de documentación.
Inconvenientes: Falta de comprensión del **contexto**, necesidad de **supervisión** humana, y problemas de **seguridad** y **propiedad intelectual**.

Limitaciones de las Herramientas CASE Actuales:

UML no tiene una **gramática formal,** lo que complica la traducción directa de diagramas a código fuente sin un esquema intermedio como XMI *(OMG,2017).*

03 – Objetivos y Metodología.

Objetivo General:

Desarrollar una herramienta CASE que permita a los estudiantes de **DAM y DAW** crear diagramas UML y generar automáticamente **código Java**, facilitando el aprendizaje práctico de la **POO**.

Objetivos Específicos:

Implementar una **aplicación web** con una **GUI** intuitiva y fácil de usar para **crear diagramas UML**.
Incorporar a la **aplicación web** la funcionalidad de **convertir diagramas UML a XMI**.
Desarrollar un **parser** en Python que analice el documento **XMI** y genere hechos y reglas de **CLIPS**.
Evaluar la herramienta mediante **pruebas** y realizar ajustes según el feedback.

03 – Objetivos y Metodología.

Metodología del Trabajo:

Metodología Ágil (Kanban): Gestión visual de tareas con enfoque iterativo para garantizar flexibilidad y adaptación a los cambios (**priorizar**).
Uso del sistema de control de vesiones **Git**.

04 – Desarrollo.

Tecnologías Utilizadas en el desarrollo de la herramienta CASE:

HTML y JavaScript: para creación de diagramas UML y conversión a XMI.
Python: Para leer archivos XMI y convertirlos en hechos para CLIPS.
CLIPS: Para la generación de código Java a partir de los hechos derivados de UML.
XMI: Formato estándar para la interoperabilidad entre herramientas CASE.
Java: Lenguaje de programación para el código generado.

Requisitos Funcionales :

El código generado incluya las clases, **atributos** y métodos para relaciones de **herencia y asociaciones dirigidas** (tanto reflexivas como binarias).

Requisitos No Funcionales :

Usabilidad, rendimiento, escalabilidad y fácil de mantener.

Evaluación :

Se verificó la funcionalidad de cada componente y se validó el código Java generado.

04 – Desarrollo.
Aplicación Web

Implementada en JavaScript su GUI permite crear los diagramas UML sobre el canvas.

GUI de la aplicación web.

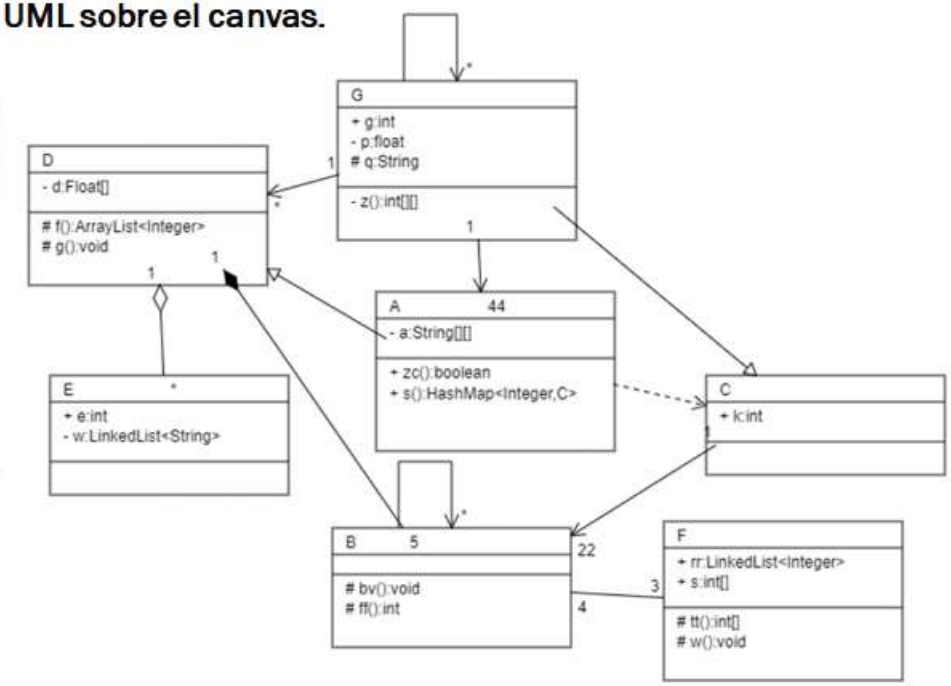


Diagrama UML realizado con la herramienta.

04 – Desarrollo.
Aplicación Web

El **script.js** tiene dos funcionalidades principales: 1 Crear el diagrama UML y 2 Exportarlo a XMI.

1 Creación del diagrama UML

Cada diagrama consta de dos arrays de objetos:

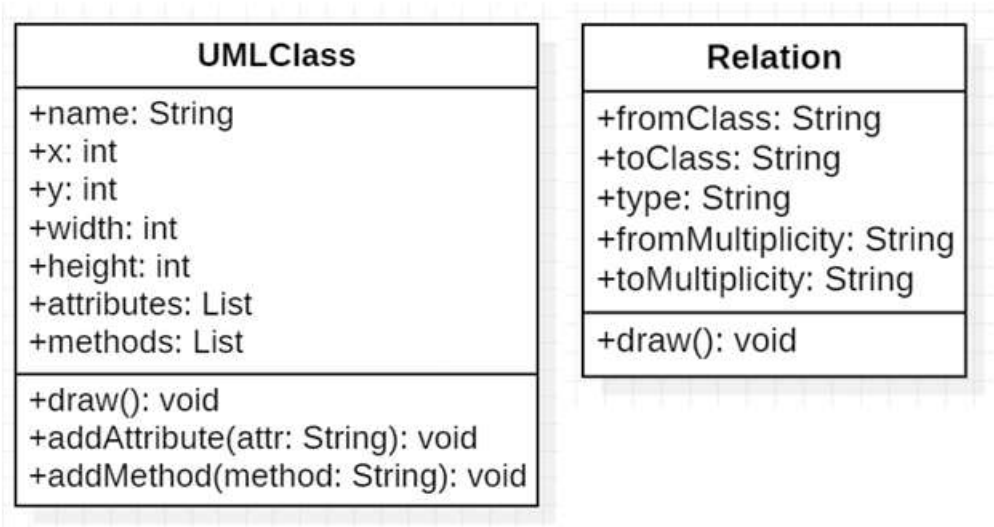
```
const classes = [];
const relations = [];
```

Y una función llamada:

```
drawDiagram()
```

Las clases tienen atributos y métodos:

```
const attr = `${visibility} ${attribute}:${type}`;
const meth = `${visibility} ${method}():${type}`;
```



Cada vez que se desplaza una clase o se crea una clase, atributo, método o relación se llama a drawDiagram().

04 – Desarrollo.
Aplicación Web

2 Exportación de UML a XMI

Toda la información del diagrama UML está almacenado en los dos arrays de objetos:

```
const classes = [];
const relations = [];
```

Prólogo

```
<?xml version="1.0" encoding="UTF-8"?>
  <XMI xmi.version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:uml="http://www.omg.org/spec/UML/20090901">
  <uml:Model xmi:type="uml:Model" name="UMLModel">
```

Definición de clases

```
<packagedElement xmi:type="uml:Class" name="b">
  <ownedAttribute visibility="-" name="b1" type="String" />
  <ownedOperation visibility="#" name="g" type="boolean[]" />
  <ownedOperation visibility="-" name="nc" type="Integer" />
</packagedElement>
<packagedElement xmi:type="uml:Class" name="c">
  <ownedOperation visibility="-" name="w" type="ArrayList&lt;Float&gt;" />
</packagedElement>
```

Definición de relaciones

```
<packagedElement xmi:type="uml:Generalization" memberEnd="a b">
</packagedElement>
<packagedElement xmi:type="uml:DirectedAssociation" memberEnd="a c">
  <ownedEnd type="a" multiplicity1="1" />
  <ownedEnd type="c" multiplicity2="*" />
</packagedElement>
```

04 – Desarrollo.
Parser en Python

Traductor.py recorre el árbol XMI

Lee el archivo XMI y genera los facts para CLIPS.
import xml.etree.ElementTree as ET

Algunas funciones importantes son:

parse_xmi(file_path)

Obtiene el elemento **raíz** del árbol XML.

extract_classes(root)

Almacena las clases en clases_dic ={}

extract_directed_associations(root, class_dict)

Añade un atributo a la clase origen de tipo clase destino (pudiendo ser un array de tamaño fijo o variable)

extract_generalizations(root)

...

```python
import xml.etree.ElementTree as ET

def parse_xmi(file_path):
    tree = ET.parse(file_path)
    root = tree.getroot()
    print(f"Root element: {root.tag}")
    return root

def extract_classes(root):
    classes = []
    class_dict = {}
    for elem in root.findall('.//packagedElement'):
        type_attr = elem.get('{http://schema.omg.org/spec/XMI/2.1}type')
        if type_attr == 'uml:Class':
            class_name = elem.get('name')
            class_info = {
                'name': class_name,
                'attributes': [],
                'operations': []
            }
            class_dict[class_name] = class_info
            # Extraer atributos de la clase
            for attr in elem.findall('ownedAttribute'):
                attr_name = attr.get('name')
                attr_visibility = attr.get('visibility')
                attr_type = attr.get('type')
                if attr_visibility == "+":
```

```python
# Añadir atributo en la clase source
class_name = source
if class_name in class_dict:
    if multiplicity_target != "*":
        class_dict[class_name]['attributes'].append({
            'name': f'{target.lower()}List{obj_id}',
            'visibility': 'private',
            'type': f'{target}[]'
        })
        obj_id+=1
    else:
        class_dict[class_name]['attributes'].append({
            'name': f'{target.lower()}List{obj_id}',
            'visibility': 'private',
            'type': f'HashSet<{target}>'
        })
        obj_id+=1
```

04 – Desarrollo.
Parser en Python

Traductor.py traducción a hechos de CLIPS.

generate_clips_facts(classes, relationships)

Genera los hechos en el formato CLIPS a partir de las clases y relaciones extraídas del archivo XMI.

```
(deffacts initial-facts
(attribute (id attr1) (class-name A) (name a) (visibility protected) (type "float"))
(attribute (id attr2) (class-name A) (name bList1) (visibility private) (type "HashSet<B>"))
(attribute (id attr3) (class-name A) (name aList2) (visibility private) (type "A[]"))
(operation (id op1) (class-name A) (name x) (visibility private) (type "ArrayList<Integer>"))
(directedAssociation (source A) (target B) (multiplicity1 1) (multiplicity2 *))
(directedAssociation (source A) (target A) (multiplicity1 1) (multiplicity2 5))
(class (name A) (attributes attr1 attr2 attr3) (operations op1))
...
```

04 – Desarrollo.
Parser en Python

Traductor.py definición de reglas de CLIPS.

```
(class (name A) (attributes attr1 attr2 attr3) (operations op1))
(generalization (child A) (parent K))
(attribute (id attr1) (class-name A) (name a) (visibility protected) (type "float"))
(attribute (id attr2) (class-name A) (name bList1) (visibility private) (type "HashSet<B>"))
(operation (id op1) (class-name A) (name x) (visibility private) (type "ArrayList<Integer>"))
```

```
(defrule generate-java-code
   ?class <- (class (name ?class-name) (attributes $?attributes) (operations $?operations))
   (generalization (child ?class-name) (parent ?x))
=>
   (printout t "// Java code for class " ?class-name crlf)
   (printout t "public class " ?class-name " extends " ?x " {" crlf)
 ;; Imprimir atributos
   (do-for-all-facts ((?attr attribute))
   (and
   (member$ (fact-slot-value ?attr id) $?attributes)
   (eq (fact-slot-value ?attr class-name) ?class-name))
   (bind ?visibility (fact-slot-value ?attr visibility))
   (bind ?type (fact-slot-value ?attr type))
   (bind ?name (fact-slot-value ?attr name))
   (printout t " " ?visibility " " ?type " " ?name ";" crlf))
 ;; Imprimir operaciones
   (do-for-all-facts ((?op operation))
   (and
   (member$ (fact-slot-value ?op id) $?operations)
   (eq (fact-slot-value ?op class-name) ?class-name))
   (bind ?visibility (fact-slot-value ?op visibility))
   (bind ?type (fact-slot-value ?op type))
   (bind ?name (fact-slot-value ?op name))
   (printout t " " ?visibility " " ?type " " ?name "()" " {" crlf
   " // method body" crlf " }" crlf))
   (printout t "}" crlf crlf)
)
```

04 – Desarrollo.
CLIPS

Generación de código Java.

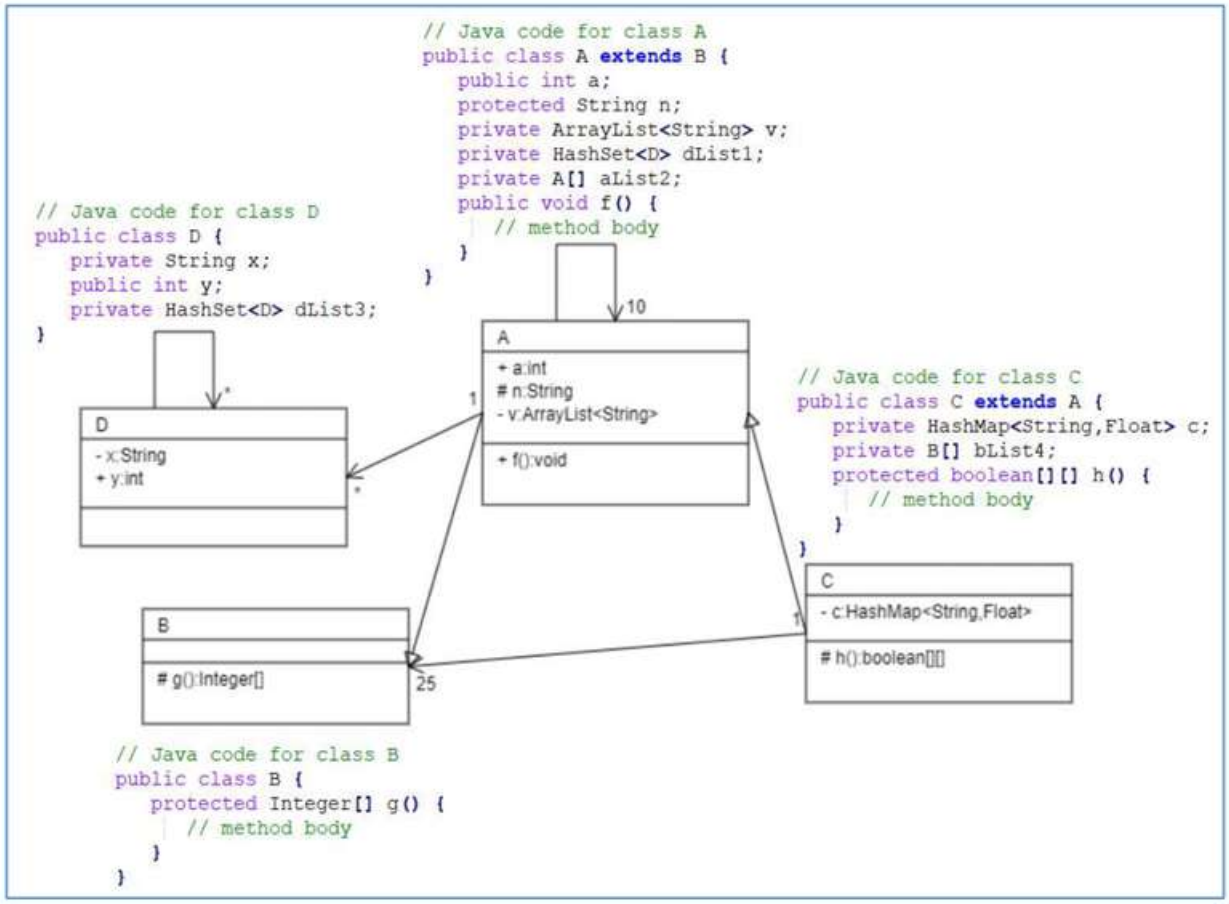
CLIPS IDE

File Edit Environment Debug Help

Dir: C:\Users\carlo\OneDrive\Desktop\TFE

```
         CLIPS (6.4 2/9/21)
CLIPS> (clear)
CLIPS> (load "facts.clp")
%%%%%%%%%%**
TRUE
CLIPS> (reset)
CLIPS> (run)
// Java code for class G
public class G extends C {
   public int g;
   private float p;
   protected String q;
   private HashSet<D> dList3;
   private A[] aList4;
   private HashSet<G> gList5;
   private int[][] z() {
      // method body
   }
}

// Java code for class A
public class A extends D {
   private String[][] a;
   public boolean zc() {
```



05 – Conclusiones y Trabajo Futuro.

Conclusiones

Desarrollo exitoso de la herramienta CASE:

Evaluación Positiva: La herramienta cumple con los requisitos funcionales y no funcionales.

Facilita el aprendizaje y desarrollo de software para estudiantes de FP, demostrando la integración efectiva de tecnologías Web, Python y Sistemas Expertos.

Resultado 1 – Creación de Diagramas UML.
Resultado 2 – Exportación a Formato XMI.
Resultado 3 – Parsing de Archivos XMI.
Resultado 4 – Generación Automática de Código Java.
Resultado 5 – Reconocimiento de Patrones de Diseño Software.

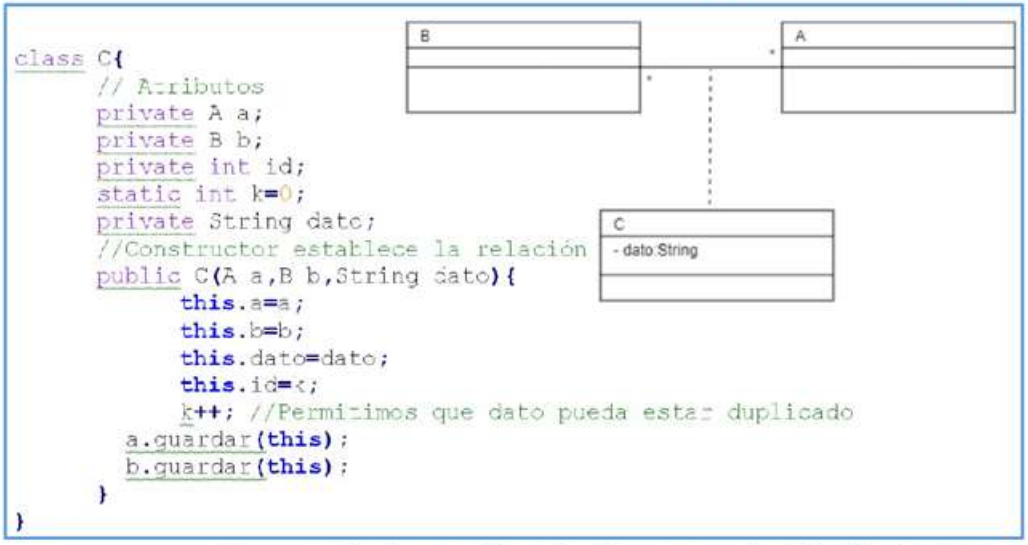
La siguiente regla en CLIPS reconoce en el diagrama el patrón State.

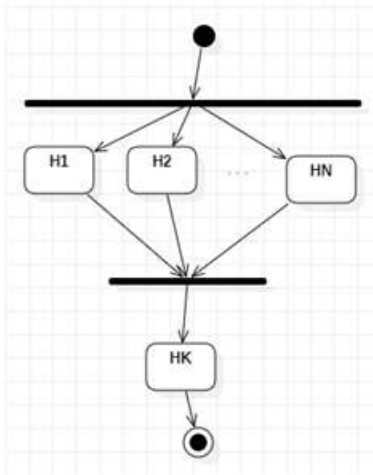
```
CLIPS> ;; Regla para identificar el patrón de diseño State
(defrule identify-state-pattern
(composition (whole ?context) (part ?state))
?stateClass <- (class (name ?state) (attributes $?attributes) (operations $?operations))
?contextClass <- (class (name ?context))
(test (neq ?context ?state)) ;; Evitar que la clase contexto sea la misma que la clase estado
(exists (generalization (child ?state)))
=>
(printout t "Se ha identificado un patrón de diseño State:" crlf)
(printout t "Clase Contexto: " ?context crlf)
(printout t "Clase Estado: " ?state crlf)
(printout t "Clases que heredan de " ?state ":" crlf)
(do-for-all-facts ((?gen generalization))
(eq (fact-slot-value ?gen parent) ?state)
(printout t " " (fact-slot-value ?gen child) crlf))
)
```

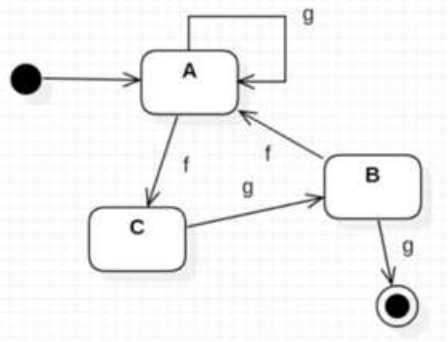
05 – Conclusiones y Trabajo Futuro.
Trabajo Futuro

Extensión a Otros Lenguajes.
Mejoras en la GUI de la aplicación web (eliminación de atributos, relaciones etc.).
Incorporación de Diagramas UML Adicionales (se ha indicado en la memoria cómo codificarlos).
Clase asociación o los diagramas UML de comportamiento, en concreto, el de **estados** y el de **actividades**.

```
class C{
      // Atributos
      private A a;
      private B b;
      private int id;
      static int k=0;
      private String dato;
      //Constructor establece la relación
      public C(A a,B b,String dato){
            this.a=a;
            this.b=b;
            this.dato=dato;
            this.id=k;
            k++; //Permitimos que dato pueda estar duplicado
        a.guardar(this);
        b.guardar(this);
      }
}
```







Reconocimiento de más Patrones de Diseño Software.
Sugerencias de diseño.

05 – Fin