Entrega final

Camilo Bolivar

Julian Leon

Parte Teórica (Revisar el documento de referencia en el classroom para la sustentación teórica)  
\* Documento con el análisis de los casos de uso del robot (y Robot Story)

Análisis casos Robot

1. Explorar el terreno: El robot debe ser capaz de moverse de manera autónoma y explorar el entorno en busca de objetos reciclables.

2. Reconocer objetos: El robot debe utilizar una cámara y un sistema de inteligencia artificial para reconocer objetos reciclables en su entorno.

3. Agarrar objetos: El robot debe ser capaz de calcular la posición y orientación de su garra para agarrar objetos reciclables de manera precisa.

4. Depositar objetos en contenedor: El robot debe llevar los objetos reciclables a un contenedor correspondiente a su categoría y depositarlos allí.

5. Cambiar de categoría: El robot debe poder cambiar de categoría de objetos reciclables y continuar recolectando los objetos de la nueva categoría.

6. Navegación autónoma: El robot debe ser capaz de navegar de manera autónoma en terrenos planos y no uniformes, evitando obstáculos y manteniendo una distancia segura de los objetos.

7. Comunicar estado: El robot debe ser capaz de comunicar su estado actual, ya sea mediante una interfaz de usuario o mediante señales visuales

Análisis casos Robot Story

1. El robot parte desde una posición inicial llamada "home".

2. Utilizando su cámara, explora el terreno en busca de objetos reciclables.

3. Cuando encuentra un objeto, ajusta la trayectoria de su cámara para centrar el objeto en la imagen y utilizar su inteligencia artificial para reconocer la categoría del objeto.

4. Con base en la información del objeto, el robot calcula la posición y orientación de su garra para agarrar el objeto.

5. Una vez que el objeto está en su garra, el robot lo lleva al contenedor correspondiente a su categoría y lo deposita allí.

6. El robot vuelve a su posición inicial "home" para continuar la exploración en busca de más objetos.

7. Si no encuentra más objetos en su alcance, se mueve a otro lugar para realizar otra exploración.

8. Durante su funcionamiento, el robot puede enfrentar diferentes eventos, como soltar un objeto, cambiar la posición u orientación de un objeto, problemas mecánicos o eléctricos, entre otros. Estos eventos son verificados en cada estado del robot y pueden generar transiciones en su máquina de estados.

9. El robot también puede recibir comandos desde una interfaz de usuario, como la parada de emergencia, continuar, inicio, agarrar objeto, soltar objeto, cambiar categoría, entre otros.

10. El robot utiliza una interfaz de usuario para mostrar su estado actual y permitir la interacción con los comandos.  
\* Diagrama de clases del robot (genmymodel)

Código

[module uml2python('http://www.eclipse.org/uml2/4.0.0/UML')/]

[comment]

Main entry point. This main template MUST be named 'generate' and it must

own the main annotation.

[/comment]

[template public generate(m : Package)]

[comment @main/]

[file (m.name.concat('.py'), false, 'UTF-8')]

[if (not m.ownedComment->isEmpty())]

"""

[m.ownedComment.genComment(' ')/]

"""

[/if]

[comment Dummy and very simple import management (misses the extern module imports) /]

[if (not m.allOwnedElements()->filter(TypedElement)->select(type <> null and type.name <> null and type.name = 'Date')->isEmpty())]

from datetime import datetime

[/if]

[comment Generate code for classes/interfaces /]

[let nonInherit : Set(Classifier) =

m.ownedElement->

filter(Classifier)->

select(generalization->isEmpty())->reject(oclIsKindOf(Class) and not oclAsType(Class).interfaceRealization->isEmpty())]

[for (e : Classifier | nonInherit->sortedBy(not oclIsKindOf(Enumeration)))]

[e.genClassif()/]

[/for]

[for (e : Classifier | m.ownedElement->filter(Classifier)

->select(c | nonInherit->excludes(c))

->sortedBy(getAllUsedInterfaces()->union(getGenerals())->size()))]

[e.genClassif()/]

[/for]

[/let]

# [protected ('-> functions/methods for ' + m.name + ' package')]

# [/protected]

[/file]

[/template]

[comment]

These templates are used to generate 'Classifier' code, i.e., Class, Interface and Enumeration.

[/comment]

[template public genClassif(e : Classifier)/]

[template public genClassif(c : Class)]

[let inherited : Bag(Classifier) = c.superClass->union(c.interfaceRealization.contract)]

class [c.name/]([if (not inherited->isEmpty())][for (cl : Classifier | inherited) separator(', ')][cl.name/][/for][else]object[/if]):

[if (not c.ownedComment->isEmpty())]

"""

[c.ownedComment.genComment(' ')/]

"""

[/if]

[let navig : Bag(Property) = c.getAssociations().navigableOwnedEnd->select(type <> c)]

[if (not c.ownedAttribute->union(navig)->isEmpty())]

def \_init\_(self):

[c.ownedAttribute.gen()/]

[navig.gen()/]

[else]

pass

[/if][/let]

# [protected ('-> properties/constructors for ' + c.name + ' class')]

# [/protected]

[if (not c.nestedClassifier->isEmpty())]

[c.nestedClassifier.genClassif()/]

[/if]

[if (not c.ownedOperation->isEmpty())]

[for (ops : Operation | c.ownedOperation)]

[ops.gen()/]

[/for]

[/if]

# [protected ('-> methods for ' + c.name + ' class')]

# [/protected]

[/let]

[/template]

[template public genClassif(i : Interface)]

[let inherited : Bag(Classifier) = i.generalization.general]

class [i.name/]([if (not inherited->isEmpty())][for (cl : Classifier | inherited) separator(', ')][cl.name/][/for][else]object[/if]):

[if (not i.ownedComment->isEmpty())]

"""

[i.ownedComment.genComment(' ')/]

"""

[/if]

[let navig : Bag(Property) = i.getAssociations().navigableOwnedEnd->select(type <> i)]

[if (not i.ownedAttribute->union(navig)->isEmpty())]

def \_init\_(self):

[i.ownedAttribute.gen()/]

[navig.gen()/]

[else]

pass

[/if][/let]

# [protected ('-> properties/constructors for ' + i.name + ' class(interface)')]

# [/protected]

[if (not i.nestedClassifier->isEmpty())]

[i.nestedClassifier.genClassif()/]

[/if]

[if (not i.ownedOperation->isEmpty())]

[for (ops : Operation | i.ownedOperation)]

[ops.gen()/]

[/for]

[/if]

# [protected ('-> methods for ' + i.name + ' class(interface)')]

# [/protected]

[/let]

[/template]

[comment]

This template represents the choice we made about UML Enumeration

compilation to python code.

[/comment]

[template public genClassif(e : Enumeration)]

class [e.name/]:

[if (not e.ownedComment->isEmpty())]

"""

[e.ownedComment.genComment(' ')/]

"""

[/if]

[if (not e.ownedLiteral->isEmpty())]

[for (lit : EnumerationLiteral | e.ownedLiteral) separator(', ')][lit.name/][/for] = range([e.ownedLiteral->size()/])

[/if]

[/template]

[template public gen(p : Property)]

[if (not p.ownedComment->isEmpty())]

[p.ownedComment.genComment('#')/]

[/if]

self.[if (p.visibility = VisibilityKind::private)]\_[/if][p.name/] = [p.genValue()/]

[/template]

[template public gen(o : Operation)]

[o.header()/]

[if (not o.ownedComment->isEmpty())]

"""

[o.ownedComment.genComment(' ')/]

"""

[/if]

[o.bodyOperation()/]

[/template]

[comment]

Generate an operation header. If the operation visibility is set to

private, '\_\_' prefixes the operation name.

[/comment]

[template public header(o : Operation)]

def [if (o.visibility = VisibilityKind::private)]\_[/if][o.name/](self[for (param : Parameter | o.ownedParameter->excluding(o.getReturnResult())) before (', ') separator(', ')][param.name/][/for]):

[/template]

[template public bodyOperation(o : Operation)]

# [protected ('protected zone for ' + o.name + ' function body')]

[if (o.getReturnResult() <> null and o.getReturnResult().type <> null)]

return [o.getReturnResult().genValue()/]

[else]raise NotImplementedError

[/if]

# [/protected]

[/template]

[template public genValue(m : MultiplicityElement) post (trim())]

[if (m.isMany())]['['/]]

[elseif (m.oclIsKindOf(TypedElement))][m.oclAsType(TypedElement).type.genSingleValue()/]

[else]None

[/if]

[/template]

[comment]

Generate single values for methods and attributes initialization.

[/comment]

[template public genSingleValue(t : Type) ? (not t.oclIsUndefined()) post (trim())]

[if (t.name = 'String')]""

[elseif (t.name = 'UnlimitedNatural')]0L

[elseif (t.name = 'Double')]0.

[elseif (t.name = 'Real')]0.

[elseif (t.name = 'Float')]0.

[elseif (t.name = 'Long')]0L

[elseif (t.name = 'Integer')]0

[elseif (t.name = 'Short')]0

[elseif (t.name = 'Byte')]0x0

[elseif (t.name = 'ByteArray')]['['/]]

[elseif (t.name = 'Boolean')]False

[elseif (t.name = 'Date')]datetime()

[elseif (t.name = 'Char')]''

[elseif (t.oclIsKindOf(Enumeration))][if (not t.oclAsType(Enumeration).ownedLiteral->isEmpty())][t.name/].[t.oclAsType(Enumeration).ownedLiteral->at(1).name/][else]None[/if]

[elseif (t.oclIsKindOf(Classifier))]None

[else]None[/if]

[/template]

[template public genComment(c : Comment, prefix : String)]

[prefix/][c.genBody(prefix).replaceAll('\n','\n' + prefix + ' ')/]

[/template]

[template public genBody(c : Comment, prefix : String)]

[c.\_body/][if (not c.ownedComment->isEmpty())]['\n'/][prefix/] [c.ownedComment.genBody(prefix)->sep('\n ' + prefix)/][/if]

[/template]

[query public isMany(s : MultiplicityElement) : Boolean =

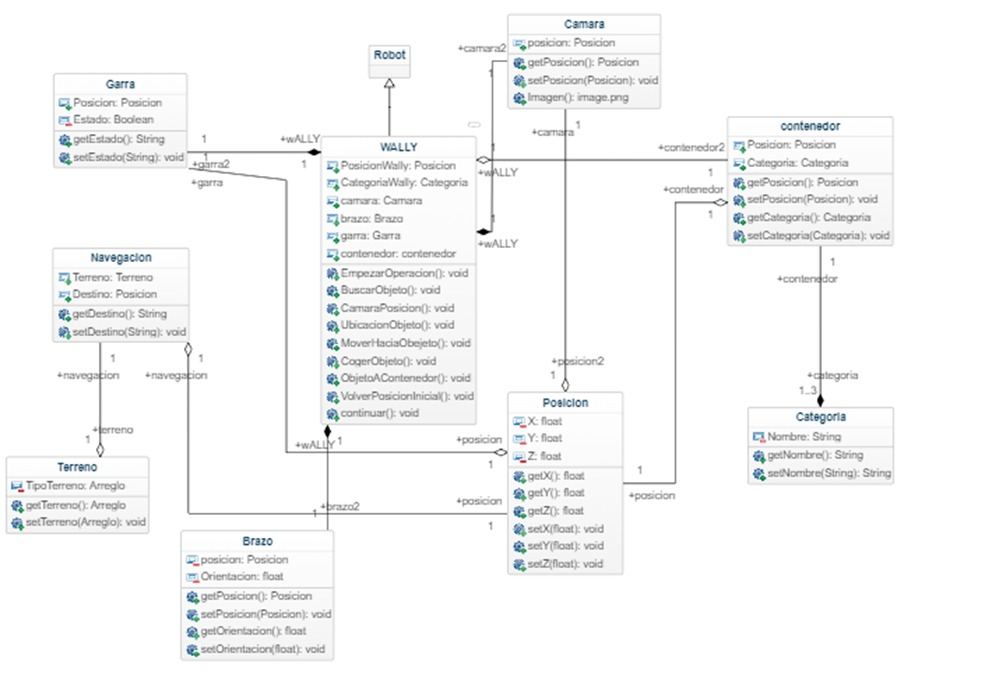
s.lower > 1 or s.upper = -1 or s.upper > 1

/]

[query public isAssociation(p : Property) : Boolean =

not p.association.oclIsUndefined()

/]



\* Diagrama UML de la máquina de estados del robot a partir de los casos de uso y los eventos.

Diagrama código robot

+-------------------+

| Explorando |

| |

| Explorar Terreno|

+--------+----------+

|

v

+-------------------+

| Reconociendo |

| Objetos |

| |

| Reconocer Objetos|

+--------+----------+

|

v

+-------------------+

| Agarrando |

| Objetos |

| |

| Agarrar Objetos |

+--------+----------+

|

v

+-------------------+

| Depositar |

| Objetos |

| |

|Depositar en |

| Contenedor |

+--------+----------+

|

v

+-------------------+

| Cambiar |

| Categoría |

| |

| Cambiar de |

| Categoría |

+--------+----------+

|

v

+-------------------+

| Navegando |

| Autónomamente |

| |

| Navegación |

| Autónoma |

+--------+----------+

|

v

+-------------------+

| Comunicando |

| Estado |

| |

| Comunicar Estado |

+-------------------+

Analisis

Explorando: El robot está explorando el terreno en busca de objetos reciclables.

Reconociendo Objetos: El robot está utilizando su cámara y sistema de inteligencia artificial para reconocer objetos reciclables en su entorno.

Agarrando Objetos: El robot está calculando la posición y orientación de su garra para agarrar objetos reciclables de manera precisa.

Depositar en Contenedor: El robot está llevando los objetos reciclables al contenedor correspondiente y los está depositando allí.

Cambiar Categoría: El robot ha cambiado de categoría de objetos reciclables y continúa recolectando los objetos de la nueva categoría.

Navegando Autónomamente: El robot está navegando de manera autónoma en terrenos planos y no uniformes, evitando obstáculos y manteniendo una distancia segura de los objetos.

Comunicando Estado: El robot está comunicando su estado actual a través de una interfaz de usuario o señales visuales

Codigo

class Robot:

def \_\_init\_\_(self, name):

self.name = name

self.position = "home"

self.category = None

self.object\_in\_grip = None

def explore\_territory(self):

# Lógica para explorar el terreno

pass

def recognize\_object(self):

# Lógica para reconocer objetos

pass

def adjust\_trajectory(self):

# Lógica para ajustar la trayectoria de la cámara

pass

def grab\_object(self):

# Lógica para agarrar un objeto

pass

def deposit\_object(self):

# Lógica para depositar un objeto en el contenedor correspondiente

pass

def return\_home(self):

# Lógica para volver a la posición inicial "home"

pass

def move\_to\_location(self, location):

# Lógica para moverse a otra ubicación

pass

def handle\_event(self, event):

# Lógica para manejar eventos inesperados

pass

def user\_interaction(self):

# Lógica para interactuar con la interfaz de usuario y recibir comandos

Pass

Diagrama story robtot

+------------------+

| Home |

| |

| Partir |

+--------+---------+

|

v

+------------------+

| Explorando |

| |

| Explorar |

| Terreno |

+--------+---------+

|

v

+------------------+

| Reconociendo |

| Objeto |

| |

| Reconocer |

| Objeto |

+--------+---------+

|

v

+------------------+

| Ajustando |

| Trayectoria |

| |

| Ajustar |

| Trayectoria |

+--------+---------+

|

v

+------------------+

| Agarrando |

| Objeto |

| |

| Agarrar |

| Objeto |

+--------+---------+

|

v

+------------------+

| Depositar |

| Objeto |

| |

|Depositar en |

| Contenedor |

+--------+---------+

|

v

+------------------+

| Home |

| |

| Volver a |

| Home |

+--------+---------+

|

v

+------------------+

| Movimiento |

| a otro lugar |

| |

| Moverse a |

| otro lugar |

+--------+---------+

|

v

+------------------+

| Eventos |

| Inesperados |

| |

| Verificar |

| Eventos |

+--------+---------+

|

v

+------------------+

| Interfaz |

| de Usuario |

| |

| Interactuar |

| con Comandos |

+------------------+

Analisis

Home: La posición inicial del robot, desde donde parte.

Explorando: El robot está explorando el terreno en busca de objetos reciclables.

Reconociendo Objeto: El robot utiliza su cámara y sistema de inteligencia artificial para reconocer la categoría de un objeto.

Ajustando Trayectoria: El robot ajusta la trayectoria de su cámara para centrar el objeto en la imagen.

Agarrando Objeto: El robot calcula la posición y orientación de su garra para agarrar el objeto.

Depositar Objeto: El robot lleva el objeto al contenedor correspondiente y lo deposita allí.

Volver a Home: El robot regresa a su posición inicial "home" para continuar la exploración.

Movimiento a otro lugar: El robot se mueve a otro lugar para realizar otra exploración.

Eventos Inesperados: El robot verifica y maneja eventos inesperados, como problemas mecánicos o eléctricos.

Interfaz de Usuario: El robot interactúa con una interf

Codigo

def run\_robot\_story():

robot = Robot("Robot1")

robot.explore\_territory()

while True:

robot.recognize\_object()

robot.adjust\_trajectory()

robot.grab\_object()

robot.deposit\_object()

robot.return\_home()

if robot.explore\_territory() == "No objects found":

robot.move\_to\_location("New location")

continue

event = check\_for\_event()

if event is not None:

robot.handle\_event(event)

command = get\_user\_command()

if command == "Stop":

break

elif command == "Continue":

continue

elif command == "Grab object":

robot.grab\_object()

elif command == "Release object":

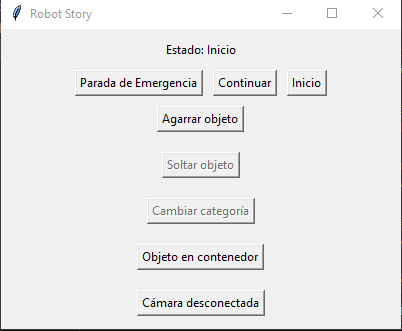
robot.release\_object()

elif command == "Change category":

robot.change\_category()

robot.user\_interaction()

\* Diseño de la interfaz de usuario gráfica (GUI), definir los frames, el layout, y los widgets



class RobotGUI(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.title("Robot Story")

self.geometry("400x300")

self.state\_label = tk.Label(self, text="Estado: Inicio")

self.state\_label.pack(pady=10)

self.buttons\_frame = tk.Frame(self)

self.buttons\_frame.pack()

self.stop\_button = tk.Button(self.buttons\_frame, text="Parada de Emergencia", command=self.stop)

self.stop\_button.grid(row=0, column=0, padx=5)

self.continue\_button = tk.Button(self.buttons\_frame, text="Continuar", command=self.continue\_execution)

self.continue\_button.grid(row=0, column=1, padx=5)

self.start\_button = tk.Button(self.buttons\_frame, text="Inicio", command=self.start)

self.start\_button.grid(row=0, column=2, padx=5)

self.grab\_object\_button = tk.Button(self, text="Agarrar objeto", command=self.grab\_object)

self.grab\_object\_button.pack(pady=10)

self.release\_object\_button = tk.Button(self, text="Soltar objeto", command=self.release\_object, state=tk.DISABLED)

self.release\_object\_button.pack(pady=10)

self.change\_category\_button = tk.Button(self, text="Cambiar categoría", command=self.change\_category, state=tk.DISABLED)

self.change\_category\_button.pack(pady=10)

self.container\_button = tk.Button(self, text="Objeto en contenedor", command=self.object\_in\_container)

self.container\_button.pack(pady=10)

self.camera\_disconnect\_button = tk.Button(self, text="Cámara desconectada", command=self.camera\_disconnect)

self.camera\_disconnect\_button.pack(pady=10)