

# Universidad Politecnica Salesiana

## Simulacion Vacunacion con Recursos

Nombre: Javier Vazquez

Materia: Simulacion

### Simpy Recursos

Utilizando las tareas de la predicci3n de llegadas de vacunas y el recinto de vacunaci3n, realizar un sistema que permita simular y correlacionar el proceso de llegada/compras de vacuna con el proceso de vacunaci3n, en donde si no se tiene un stock/número de vacunas las personas tendran que esperar/reasignar a otro día en donde exista vacunas dentro del establecimiento y realizar el proceso de vacunaci3n.

```
In [173]: import simpy
import random
import datetime as dt
from datetime import datetime
```

```
In [174]: MESAS = 10
TIEMPO_VACUNACION = 5
SEMANAS = 2
TIEMPO_SIMULACION = SEMANAS * 3 * 9 * 60
CONTROL_SIGNOS = 1
TIEMPO_POST_VACUNA = 20

tiempo_vacunacion={}
cont=0

FECHA_ACTUAL = datetime.now().strftime('%d/%m/%Y')
```

### Variables auxiliares

```
In [175]: personas_reasignadas=[]
tiempo_vacunacion={}
estado_personas=[]
```

### Definicion Clases

La primera clase que definimos es la que va a ser nuestro Container, la cual va a tener almacenada las vacunas disponibles para ser utilizadas

Con los metodos correspondiente, uno de ellos es el que va a estar constantemente verificando el estado del container y si en el caso que no encuentre stock este automaticamente llama un segundo metodo que se encarga de realizar la llamada al suministro de vacunas

```
In [176]: class VacunasEstado():
def __init__(self, env):
    self.env = env
    self.disponibles = simpy.Resource(env, capacity=MESAS)
    self.bodega = simpy.Container(env, init=350, capacity=800)
    self.monitoreo = env.process(self.monitoreo_bodega())
def monitoreo_bodega(self):
    while True:
        if self.bodega.level <= 10:
            self.logs_procesos('*****', 'Solicitar nuevas vacunas ', self.env.now)
            env.process(self.solicitar_vacunas())
def solicitar_vacunas(self):
    yield self.env.timeout(15)
def aplicar_vacuna(self):
    yield self.env.timeout(50)
    self.logs_procesos('*****', 'llega suministro vacunas ', self.env.now)
    nivel = self.bodega.capacity - self.bodega.level
    if nivel >= 0:
        nivel -= 1
    yield self.bodega.put(nivel)
def logs_procesos(self, icon, accion, hora):
    print(" %s nombre: %s) hora: %d) %s (icon, accion, hora))
```

La segunda clase definidas es de la vacunacion la cual cuenta ya con metodos realizados en anteriores tareas, como:

- control de signos vitales
- aplicacion de vacuna
- tiempo de espera post vacuna

```
In [177]: class Vacunacion():
def __init__(self, env, nombre, estadoVacunas):
    self.env = env
    self.nombre = nombre
    self.estado_vacunas = estadoVacunas
    self.proceso = env.process(self.proceso_vacunacion())
    self.personas_vacunadas = 0
def control_signos(self):
    yield self.env.timeout(random.randint(CONTROL_SIGNOS, CONTROL_SIGNOS*2))
def logs_procesos(self, icon, nombre, accion, hora):
    print(" %s nombre: %s) accion: %s) hora: %d) %s (icon, nombre, accion, hora))
def aplicar_vacuna(self):
    yield self.env.timeout(random.randint(TIEMPO_VACUNACION, TIEMPO_VACUNACION*5))
def post_vacuna(self, tiempo):
    yield self.env.timeout(tiempo)
def proceso_vacunacion(self):
    while True:
        nombre = 'persona ' + str(cont[0])
        cont[0] += 1
        self.logs_procesos('--->', nombre, " llega al recinto", self.env.now)
        estado_personas[nombre] = 'solo'
        yield env.process(self.control_signos())
        if random.randint(1, 100) > 10:
            self.logs_procesos('U', nombre, "pasa el control", self.env.now)
            estado_personas[nombre] = "pasa control"
            inicio_vacunacion = self.env.now
            if self.estado_vacunas.bodega.get(1) > 0:
                yield env.process(self.aplicar_vacuna())
                estado_personas[nombre] = "vacunado"
                self.estado_vacunas.bodega.get(1)
                tiempo_vacunacion[nombre] = self.env.now - inicio_vacunacion
                self.logs_procesos('✓', nombre, "fue vacunada ", self.env.now)
            yield env.process(self.post_vacuna(TIEMPO_POST_VACUNA))
            self.logs_procesos('X', nombre, "sin complicaciones", self.env.now)
            self.personas_vacunadas += 1
            estado_personas[nombre] = "completo"
        else:
            prox_cita = datetime.strptime(datetime.strptime(FECHA_ACTUAL, '%d/%m/%Y') + dt.timedelta(days=
            print("---XX-- No hay vacunas para %s, reasignado fecha para: %s" % (nombre, prox_cita))
            personas_reasignadas[nombre] = "reassignado"
            estado_personas[nombre] = "reassignado"
    else:
        self.logs_procesos('X', nombre, "no pasa el control", self.env.now)
        estado_personas[nombre] = "no control"
```

### Ejecutar simulacion

Primero declaramos el estado de las vacunas, el cual va a ser un proceso que esta ejecutandose de manera paralela al procesos de vacunacion, donde constantemente se encuentra verificando que exista stock de vacunas

Mientras para la vacunacion se el pasa esta clase estadoVacunas de donde podremos consumir del container(bodega vacunas) las vacunas necesarias para aplicar a los pacientes, y si en el caso que ya no exista vacunas al paciente se le reasignara la fecha para la siguiente jornada de vacunacion que es en 3 dias posteriores

```
In [178]: env = simpy.Environment()

vacunasEstado = VacunasEstado(env)
vacunacion = [Vacunacion(env, "Mesa %d" % i, vacunasEstado) for i in range(MESAS)]
env.run(until=TIEMPO_SIMULACION)
print("Simulacion realizada despues de %d semanas" % SEMANAS)
for i in vacunacion:
    print("%s) ha vacunado a [%d] personas" % (i.nombre, i.personas_vacunadas))
```



[illegible]



[illegible]



[illegible]



[illegible]



[illegible]



[illegible]