

# SUCURSAL BANCARIA

---

ENTREGA FINAL  
11.63.SIMULACIÓN

Magdalena Eppens, Sofía Gonzalez del Solar, Nicole Reiman y Francisca Sulzberger



# AGENDA

1. Caso de Negocio
2. Variables de decisión
3. Simulador
4. Experimentaciones
5. Optimización
6. Conclusión

# CASO DE NEGOCIO

# Problema a resolver



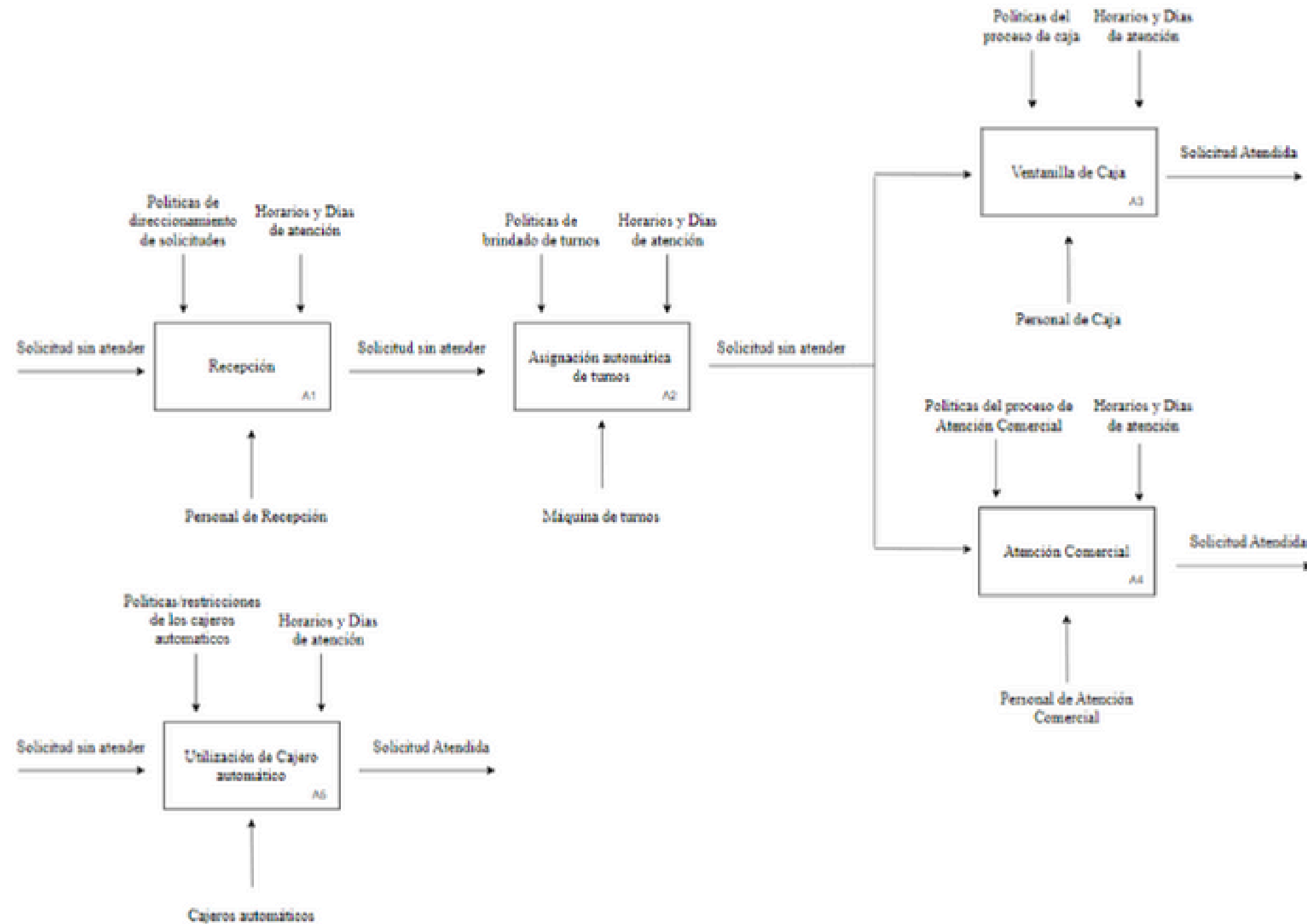
No se puede resolver mediante un sistema de turnos previo debido a que las necesidades son instantáneas e impredecibles.



Una sucursal bancaria cada día recibe numerosos clientes. La sucursal cuenta con tres servicios principales: atención comercial, ventanilla de caja y cajero automático. Este proyecto busca optimizar la atención al cliente desarrollando un modelo para reducir los tiempos de espera, enfrentando la variabilidad en la llegada de clientes, sus diferentes necesidades y los distintos niveles de experiencia de los empleados.



# Servicios modelados



# ¿Cuál es la función objetivo?



**Nivel de Servicio** = Operaciones resueltas totales / Clientes Totales

Sujeto al presupuesto



# Situación Actual



**Empleados AC = 4**

**Empleados VC = 4**

**Mantenimientos anuales CA = 6**

**Capacitaciones mensuales = 1**





# ¿A qué llamamos **operaciones resueltas**?



No todas las consultas atendidas son resueltas, esto depende de:

- El nivel de experiencia del empleado
- La cantidad de dinero disponible en el cajero automático

# Limitaciones



Presupuesto fijo mensual de **6.000 USD**

Capacidad máxima de **8** empleados totales

Mínimo de **2** empleados por sector

**No** es posible aumentar la cantidad de cajeros automáticos

¿CUÁLES SON LAS VARIABLES DE DECISIÓN?

# COSTOS VARIABLES

Arreglos de  
máquinas que  
se rompieron



# COSTOS FIJOS

Sueldos empleados  
atención comercial

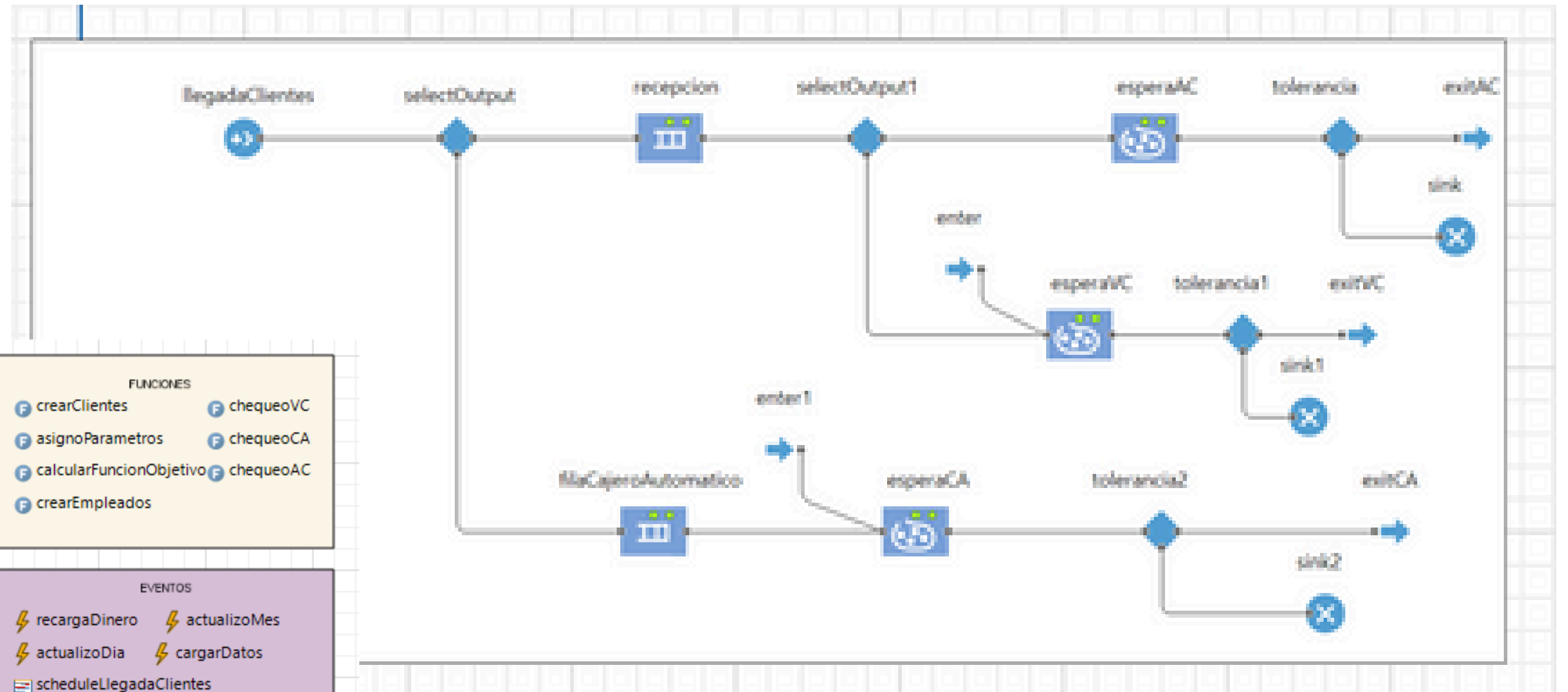
Sueldos empleados  
ventanilla de caja

Mantenimiento  
mensual



SIMULADOR

# Main



AGENTES	
empleadosAC [..]	cajeros [..]
clientes [..]	empleadosVC [..]

FUNCIONES	
crearClientes	chequeoVC
asignoParametros	chequeoCA
calcularFuncionObjetivo	chequeoAC
crearEmpleados	





EVENTOS	
recargaDinero	actualizoMes
actualizoDia	cargarDatos
scheduleLlegadaClientes	








PARAMETROS	
topeEmpleados	presupuestoMensual
cantidadAC	totalEmpleados
cantidadVC	totalMensualAC
cantidadCA	totalMensualVC
cantidadDinero	costoVC
vecesAnualesMantenimiento	costoAC
vecesCapacitacion	

VARIABLES	
indiceCA	presupuestoActual
indiceAC	operacionesResueltas
indiceVC	clientesIrritados
funcionObjetivo	clientesBase
dia	mes
movimientosVCaAC	movimientosACaVC

FUNCION OBJETIVO		
clientesRecibidosAC	clientesRecibidosCA	clientesRecibidosVC
operacionesAtendidasAC	operacionesAtendidasCA	operacionesAtendidasVC
operacionesResueltasAC	operacionesResueltasCA	operacionesResueltasVC
enCapacitacion		

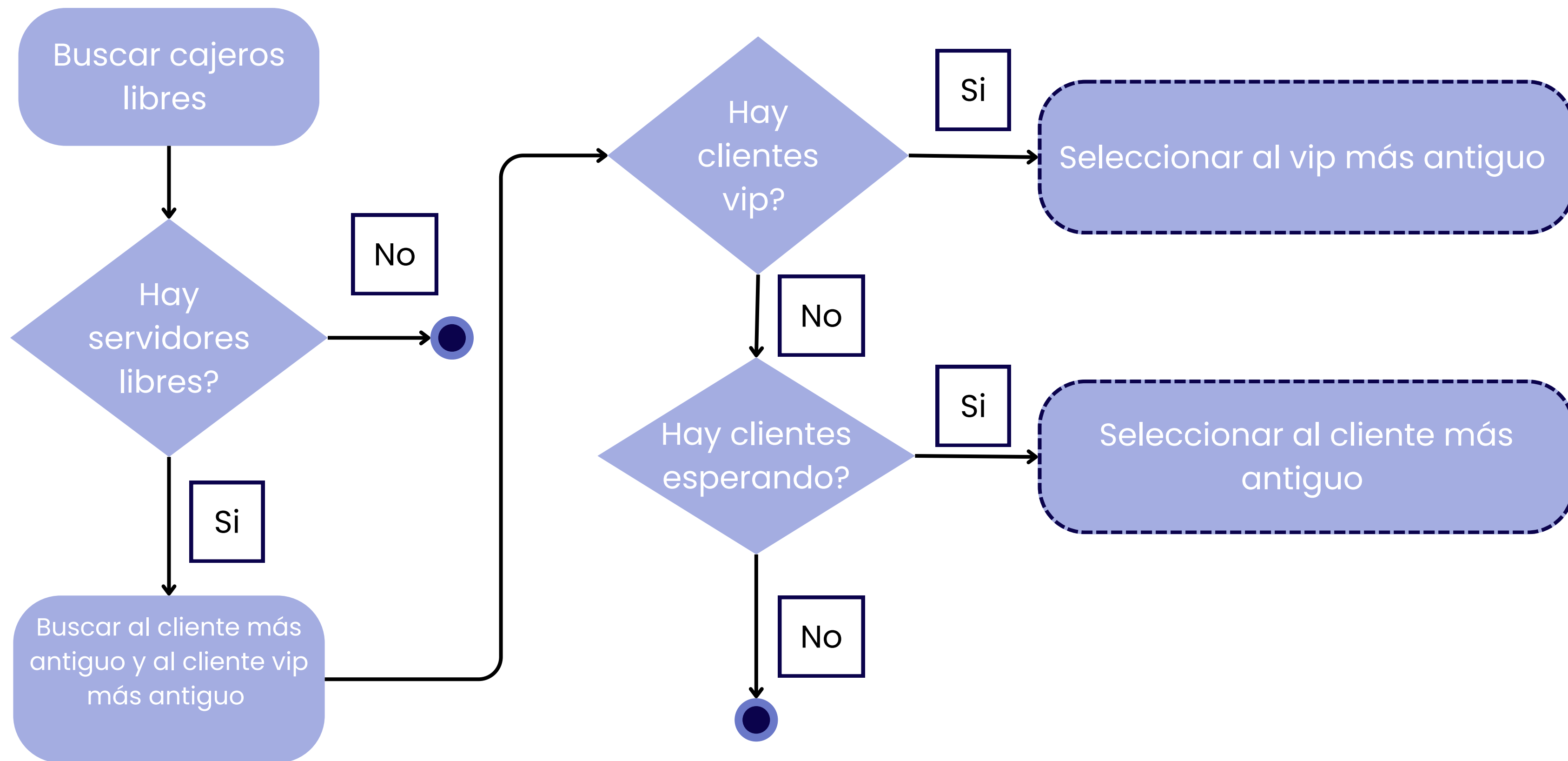
# Cliente

 tipoConsulta  
 tipoSubconsulta  
 tipoCliente  
 dificultadConsulta

 cantQuejas       Tolerancia  
 atendido       meFui  
 resuelto  
 inicioEspera  
 indice

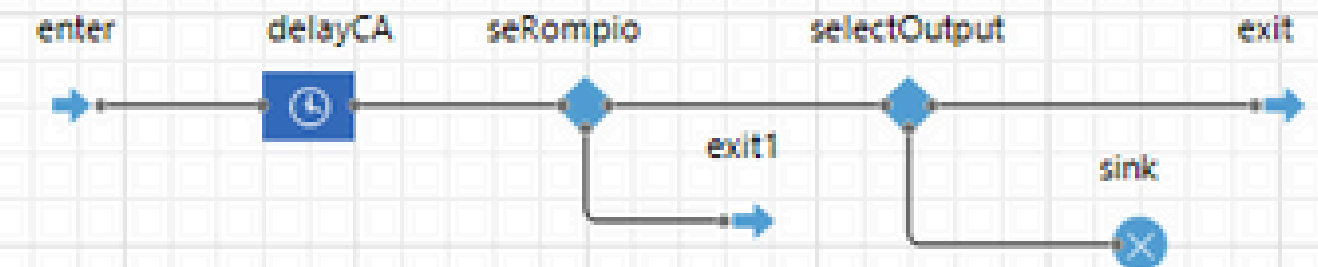
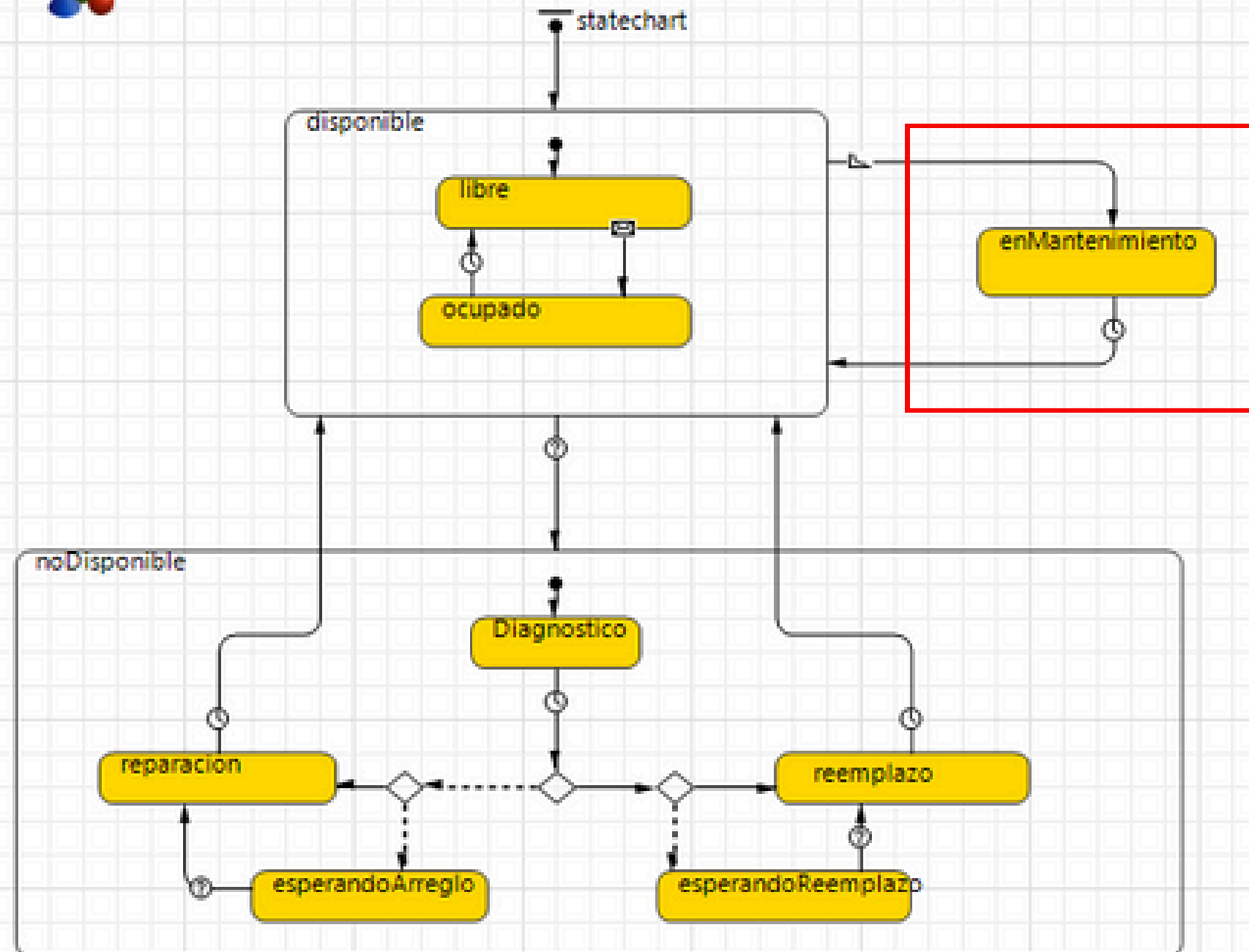


# Función para asignar clientes (se replica para todos los servicios)



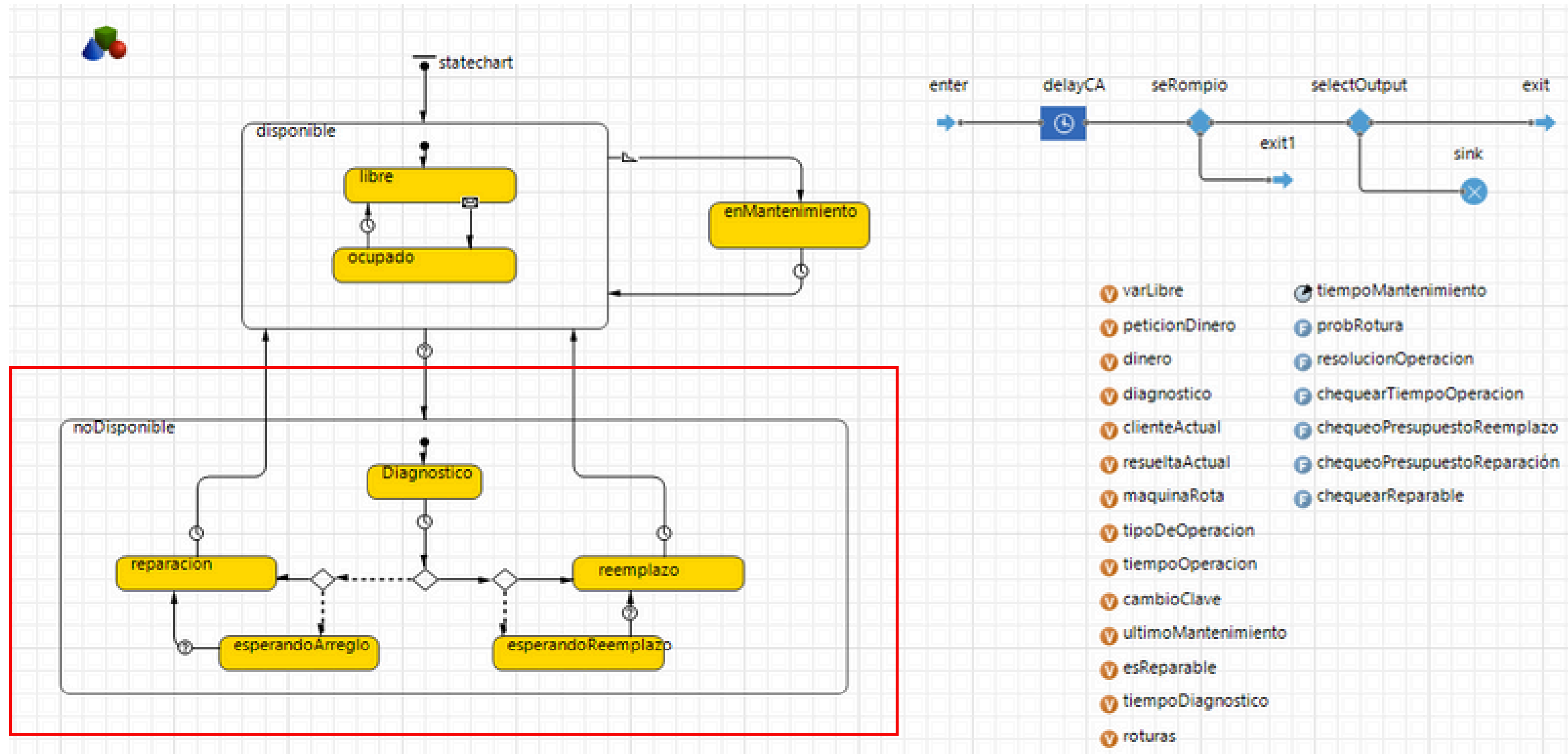
Empleado AC

# Cajero Automático

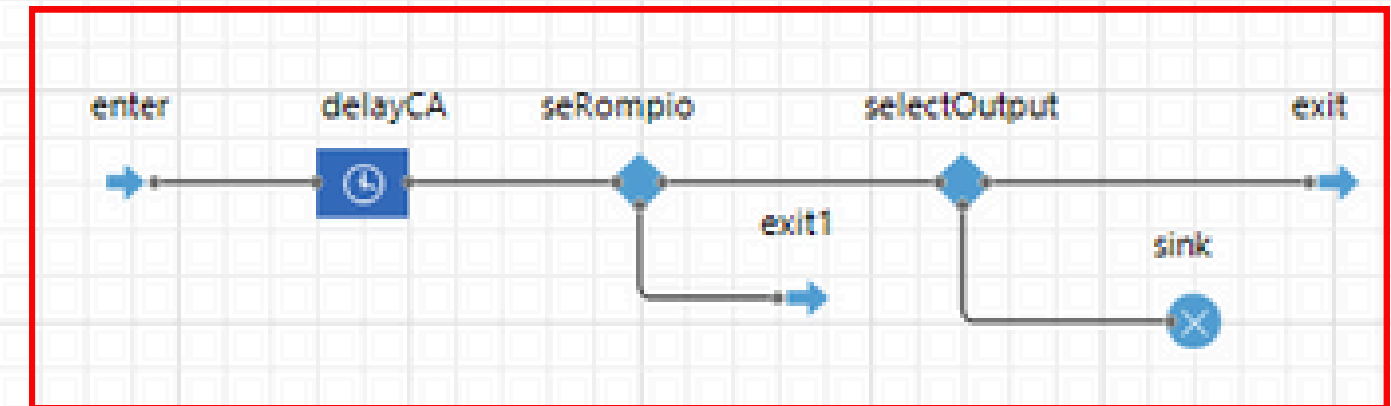
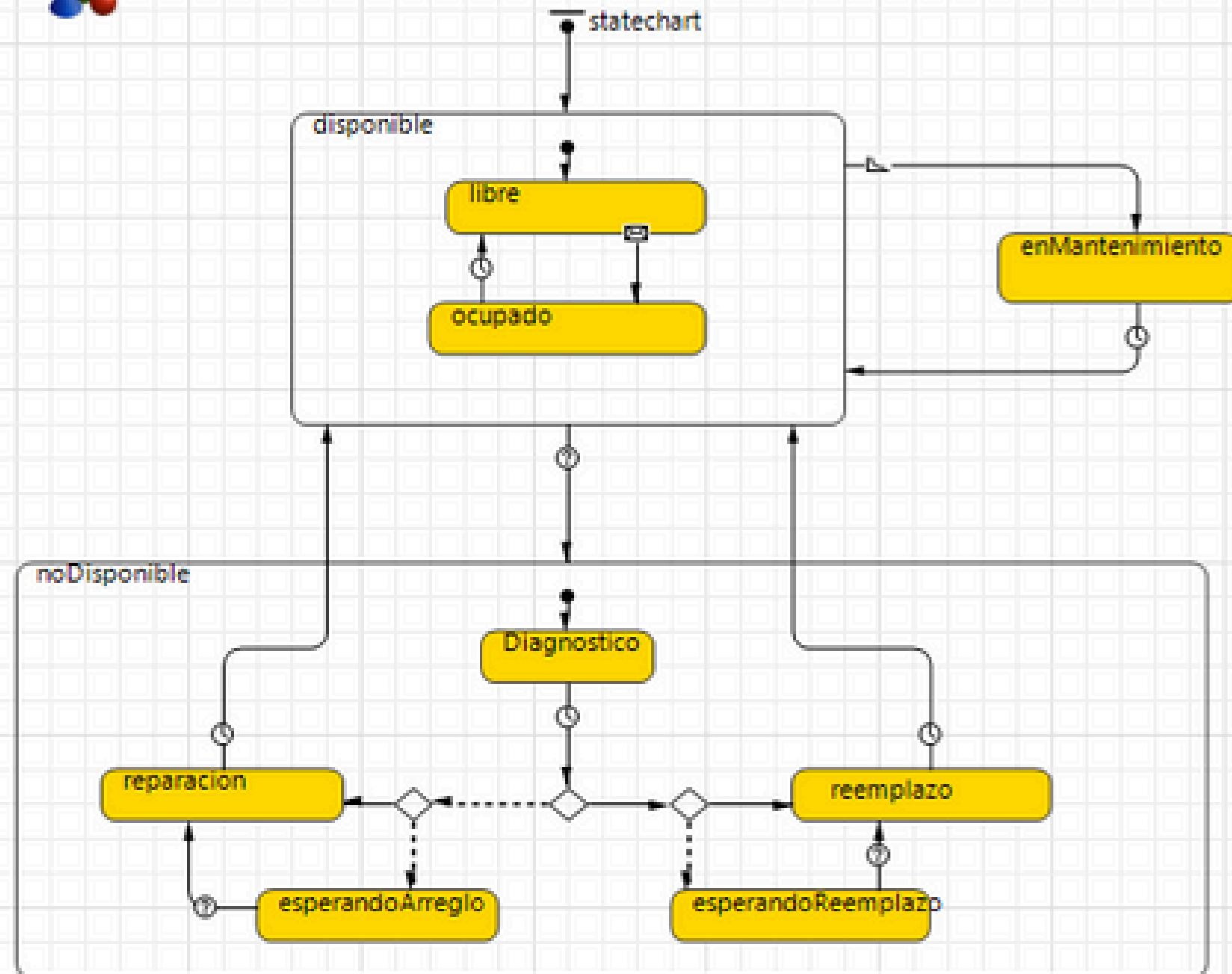


- |                     |                              |
|---------------------|------------------------------|
| varLibre            | tiempoMantenimiento          |
| peticionDinero      | probRotura                   |
| dinero              | resolucionOperacion          |
| diagnostico         | chequearTiempoOperacion      |
| clienteActual       | chequeoPresupuestoReemplazo  |
| resueltaActual      | chequeoPresupuestoReparación |
| maquinaRota         | chequearReparable            |
| tipoDeOperacion     |                              |
| tiempoOperacion     |                              |
| cambioClave         |                              |
| ultimoMantenimiento |                              |
| esReparable         |                              |
| tiempoDiagnostico   |                              |
| roturas             |                              |

# Cajero Automático

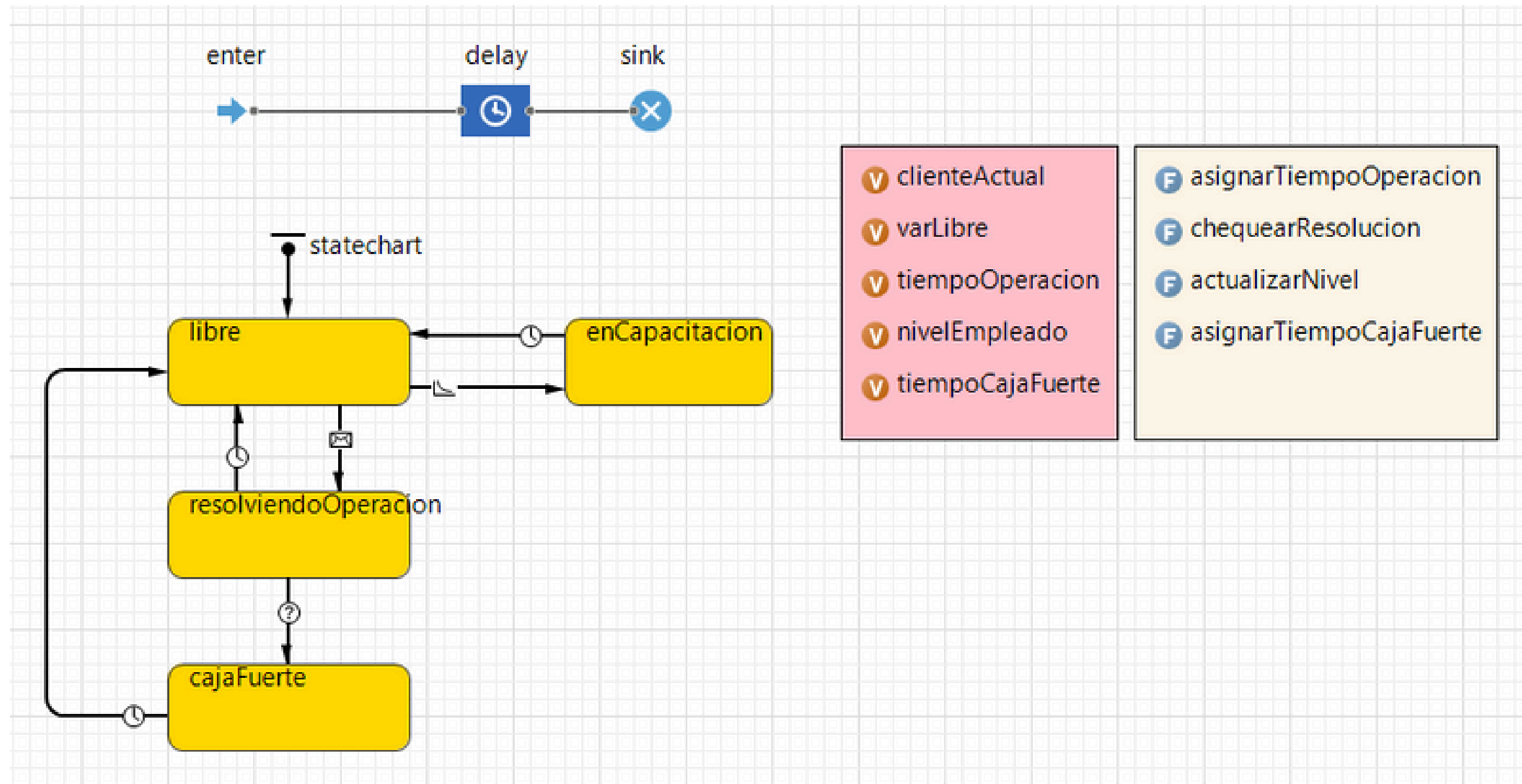


# Cajero Automático

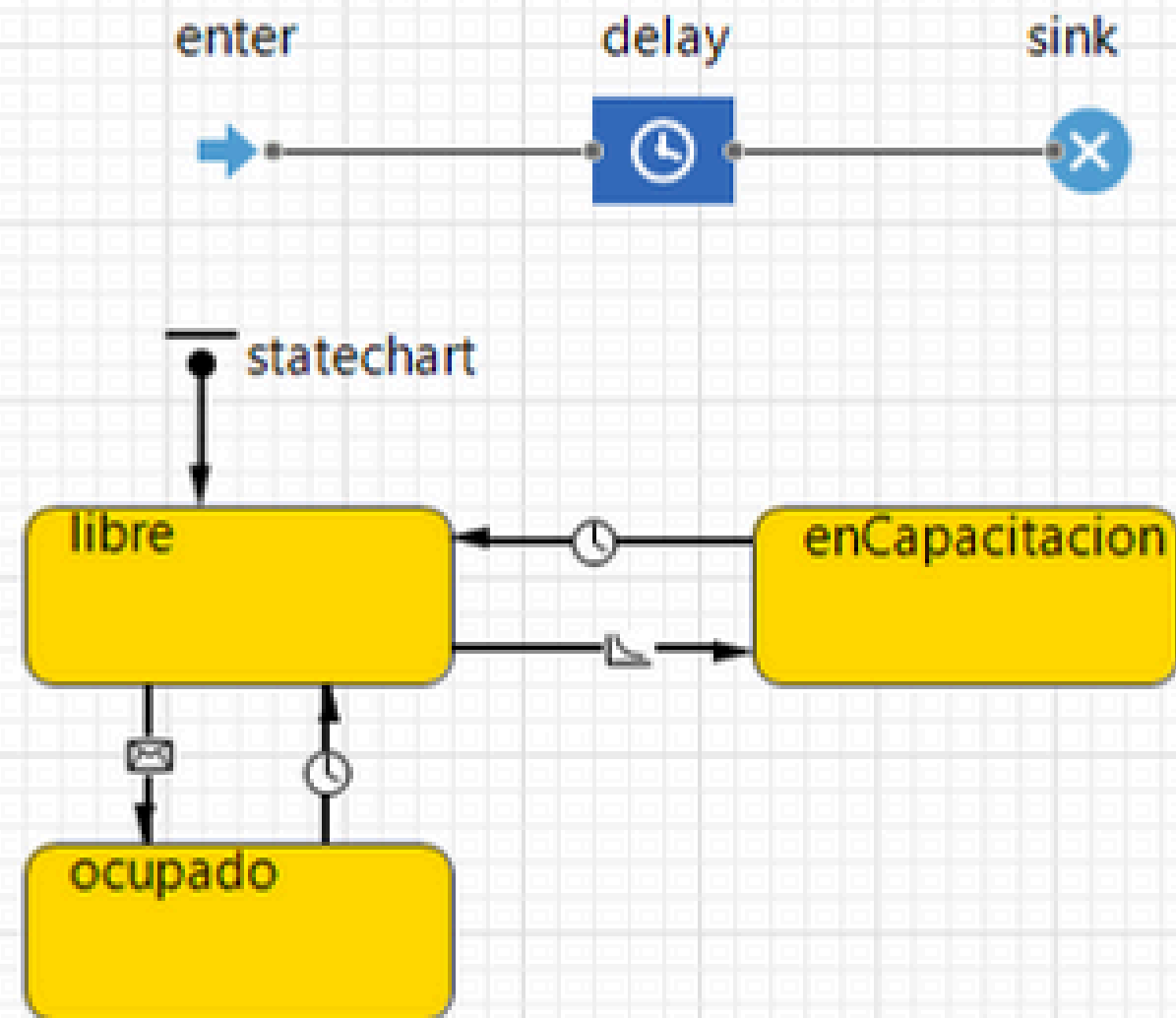


- varLibre
- peticionDinero
- dinero
- diagnostico
- clienteActual
- resueltaActual
- maquinaRota
- tipoDeOperacion
- tiempoOperacion
- cambioClave
- ultimoMantenimiento
- esReparable
- tiempoDiagnostico
- roturas
- tiempoMantenimiento
- probRotura
- resolucionOperacion
- chequearTiempoOperacion
- chequeoPresupuestoReemplazo
- chequeoPresupuestoReparacion
- chequearReparable

# Empleado AC



# Empleado VC



**v** varLibre  
**v** nivelEmpleado  
**v** clienteActual  
**v** TiempoOperacion

**F** chequearResolucion  
**F** actualizarNivel  
**F** asignarTiempoOperacion

# Capacitaciones

Properties ×

transition1 - Transition

Name: transition1 ☐ Show name ☐ Ignore

Triggered by: Rate ▼

Rate: ↺ main.vecesCapacitacion per month ▼

Action: varLibre=false;  
main.enCapacitacion=true;

Guard: main.enCapacitacion=false;

▼ Description

Properties ×

transition2 - Transition

Name: transition2 ☐ Show name ☐ Ignore

Triggered by: Timeout ▼

Timeout: ↺ 1 days ▼

Action: varLibre=true;  
main.enCapacitacion=false;

Guard:

▼ Description

## ▼ Function body

```
double random=uniform(0,1);  
  
if(random<=0.8 && nivelEmpleado<6){  
    nivelEmpleado=nivelEmpleado+1;  
}
```



# Consultas Resueltas

## ▼ Function body

```
double random=uniform(0,1);

if(clienteActual.dificultadConsulta==1 && random<0.94){
main.operacionesResueltasVC++;
clienteActual.resuelto=true;
main.operacionesResueltas++;
}

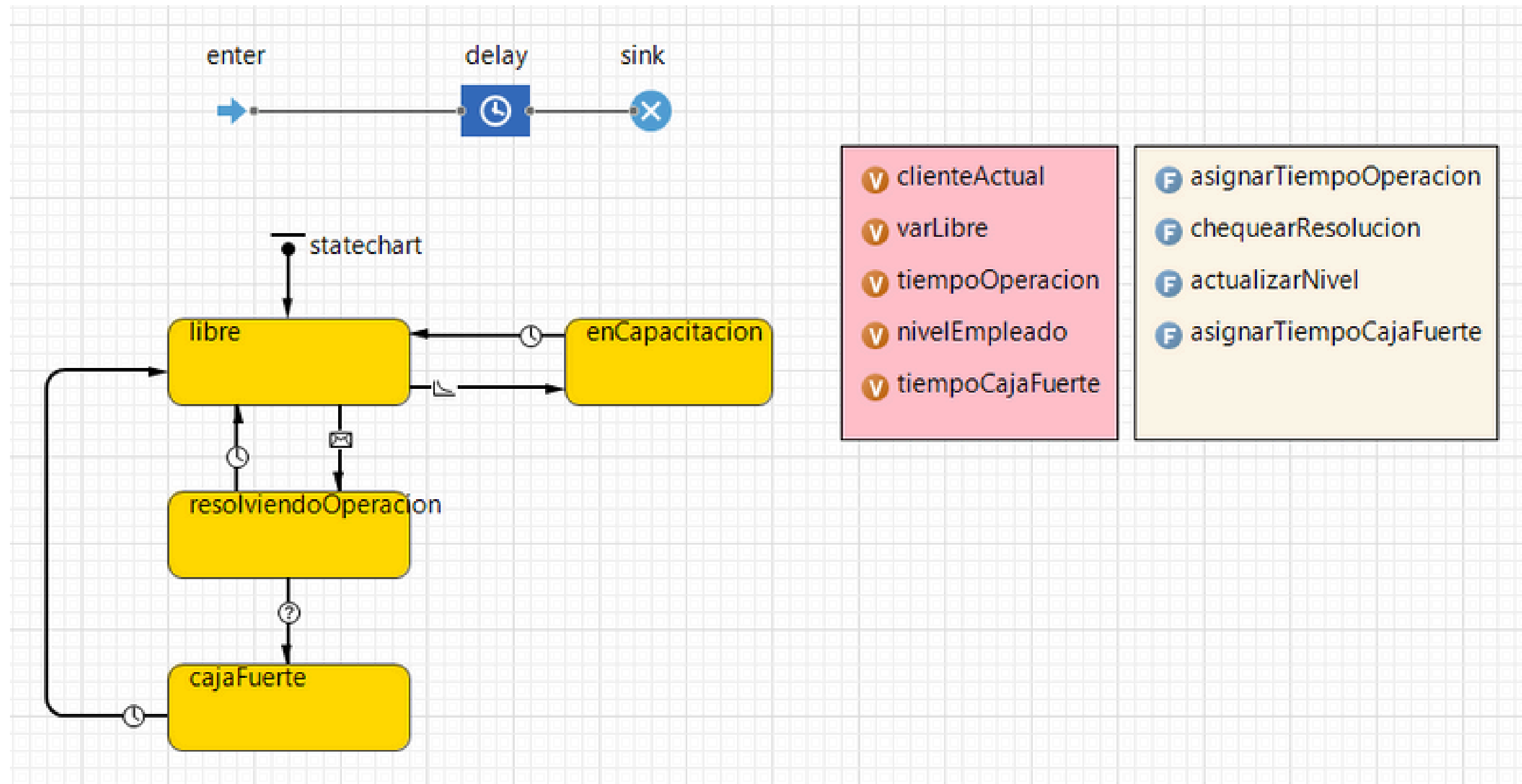
if(clienteActual.dificultadConsulta==2 && random<0.91){
main.operacionesResueltasVC++;
clienteActual.resuelto=true;
main.operacionesResueltas++;
}

if(clienteActual.dificultadConsulta==3 && random<0.86){
main.operacionesResueltasVC++;
clienteActual.resuelto=true;
main.operacionesResueltas++;
}

if(clienteActual.dificultadConsulta==4 && random<0.84){
main.operacionesResueltasVC++;
clienteActual.resuelto=true;
main.operacionesResueltas++;
}

if(clienteActual.dificultadConsulta==5){
main.operacionesResueltasVC++;
clienteActual.resuelto=true;
main.operacionesResueltas++;
}
```

# Empleado AC



# Función asignar tiempo operación AC

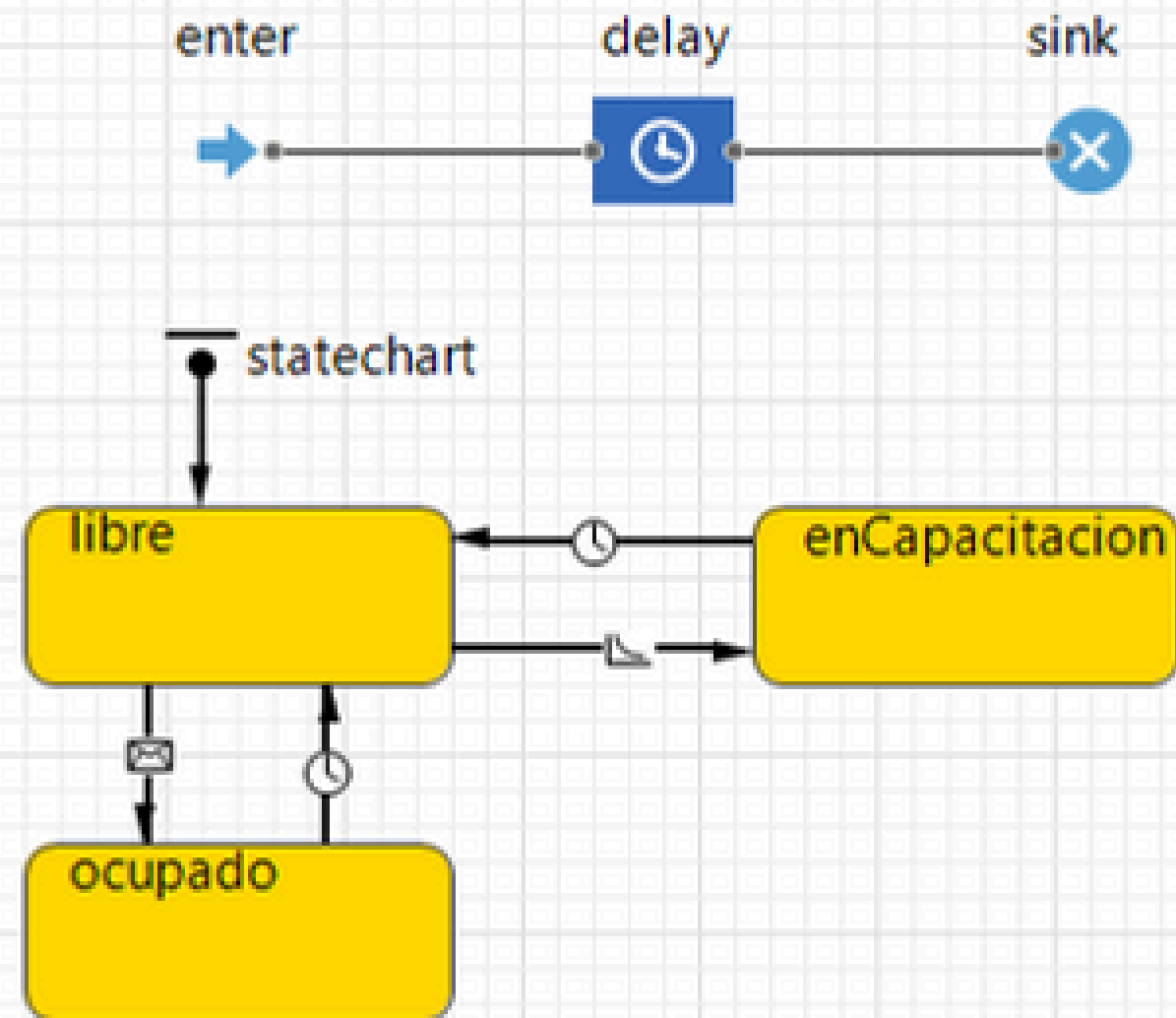
## ▼ Function body

```
if (nivelEmpleado == 4 && clienteActual.dificultadConsulta == 1) {  
    tiempoOperacion = triangular(2, 9, 6);  
}  
else if (nivelEmpleado == 4 && clienteActual.dificultadConsulta == 2) {  
    tiempoOperacion = triangular(4, 15, 7);  
}  
else if (nivelEmpleado == 4 && clienteActual.dificultadConsulta == 3) {  
    tiempoOperacion = triangular(10, 24, 13);  
}  
else if (nivelEmpleado == 4 && clienteActual.dificultadConsulta == 4) {  
    tiempoOperacion = triangular(8, 36, 20);  
}  
else if (nivelEmpleado == 5 && clienteActual.dificultadConsulta == 1) {  
    tiempoOperacion = triangular(1, 7, 4);  
}  
else if (nivelEmpleado == 5 && clienteActual.dificultadConsulta == 2) {  
    tiempoOperacion = triangular(3, 12, 5);  
}  
else if (nivelEmpleado == 5 && clienteActual.dificultadConsulta == 3) {  
    tiempoOperacion = triangular(8, 19, 11);  
}  
else if (nivelEmpleado == 5 && clienteActual.dificultadConsulta == 4) {  
    tiempoOperacion = triangular(8, 26, 17);  
}  
else if (nivelEmpleado == 6 && clienteActual.dificultadConsulta == 1) {  
    tiempoOperacion = triangular(1, 6, 3);  
}  
else if (nivelEmpleado == 6 && clienteActual.dificultadConsulta == 2) {  
    tiempoOperacion = triangular(2, 10, 3);  
}  
else if (nivelEmpleado == 6 && clienteActual.dificultadConsulta == 3) {  
    tiempoOperacion = triangular(7, 16, 9);  
}  
else if (nivelEmpleado == 6 && clienteActual.dificultadConsulta == 4) {  
    tiempoOperacion = triangular(8, 24, 15);  
}  
else if (nivelEmpleado == 6 && clienteActual.dificultadConsulta == 5) {  
    tiempoOperacion = triangular(10, 45, 16);  
}  
else {  
    tiempoOperacion = triangular(10, 45, 16);  
}
```

## ▼ Function body

```
if(clienteActual.tipoSubconsulta=="caja fuerte"){  
    tiempoCajaFuerte=uniform(5,15);  
}  
  
else{  
    tiempoCajaFuerte=0;  
}
```

# Empleado VC



**v** varLibre  
**v** nivelEmpleado  
**v** clienteActual  
**v** TiempoOperacion

**F** chequearResolucion  
**F** actualizarNivel  
**F** asignarTiempoOperacion

# Función asignar tiempo operación VC

## ▼ Function body

```
if (nivelEmpleado == 1){  
    TiempoOperacion = triangular(2, 10, 3.75);  
} else if (nivelEmpleado == 2){  
    TiempoOperacion = triangular(1, 7, 3);  
} else {  
    TiempoOperacion = triangular(1, 5, 2);  
}
```

# RESULTADOS EXPERIMENTACIONES

# ¿Cómo aumentar el nivel de servicio?

Hipótesis:



**+ clientes atendidos**



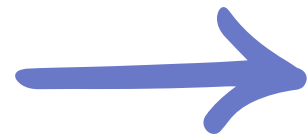
**+ consultas resueltas**

**+ nivel de experiencia del empleado**



**+ consultas resueltas**

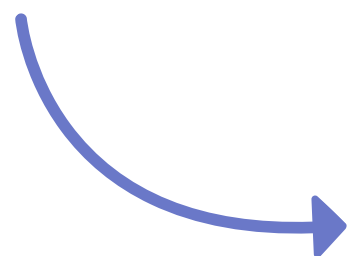
**+ empleados**



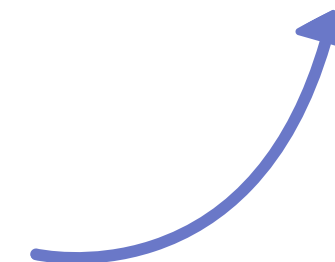
**+ consultas resueltas**

**+ mantenimientos de cajeros**

**+ consultas resueltas**



**- descomposición de cajeros**



# Costos

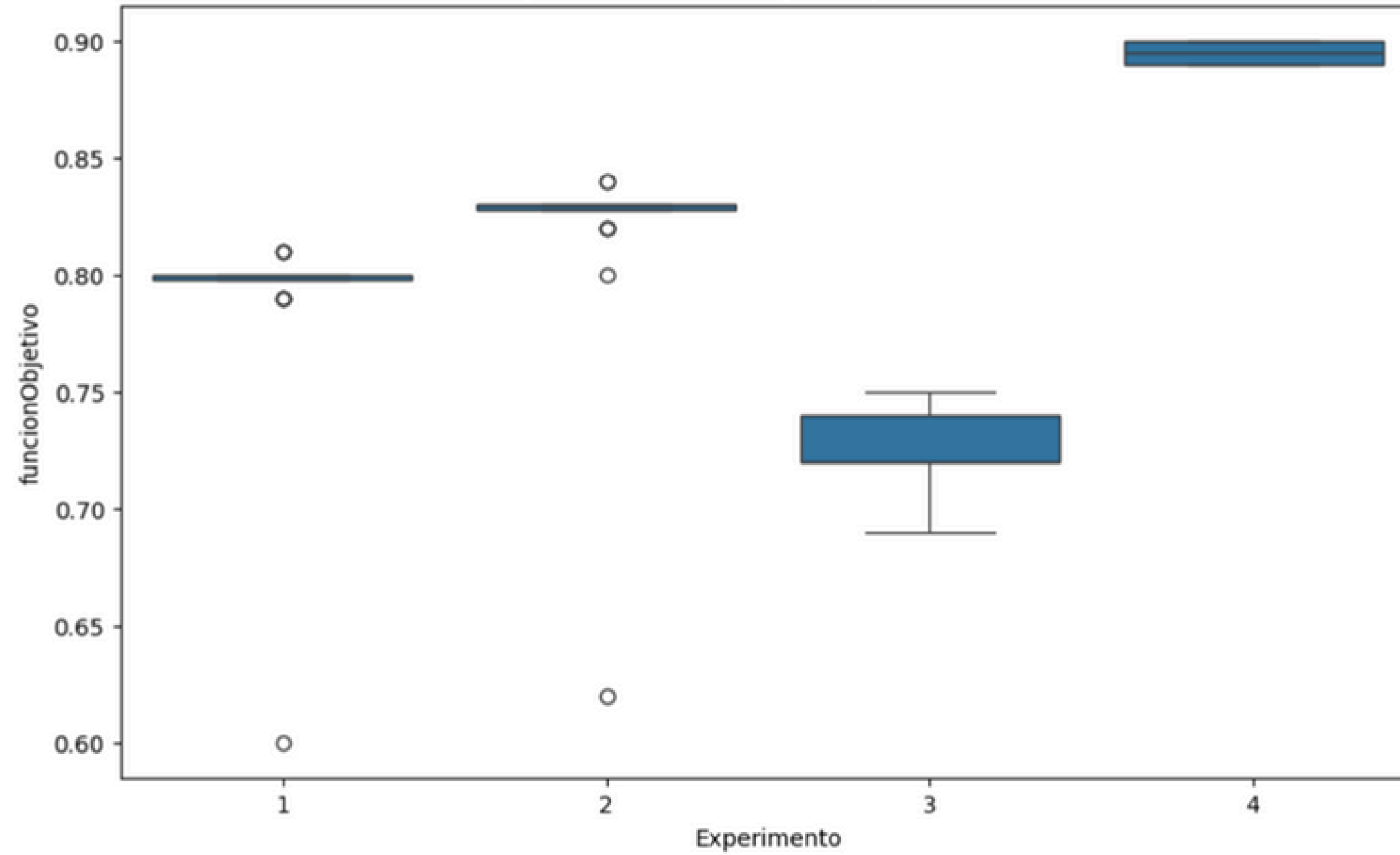
Mantenion	50
EmpleadoVC	400
EmpleadoAC	600
Mantenimiento CA	100
Arreglo CA	400
Maquina turnos	100
Reemplazo CA	2000
Empleado recepcion	300



# Experimentos

	Experimentos			
	1	2	3	4
EmpleadosAC	4	4	2	4
EmpleadosVC	4	3	2	2
Capacitaciones mensuales	1	1	1	2
Mantenimientos anuales	6	6	6	9
Costo mensual	4625	4225	2625	3937.5
Nivel de servicio	0.791	0.782	0.724	0.895
Dinero sobrante	1375	1775	3375	2062.5
Cantidad arreglos q se pueden hacer	3	4	8	5
Cantidad de reemplazos	0	0	1	1

Boxplots de funcionObjetivo por Simulador



Resultado del test de ANOVA:

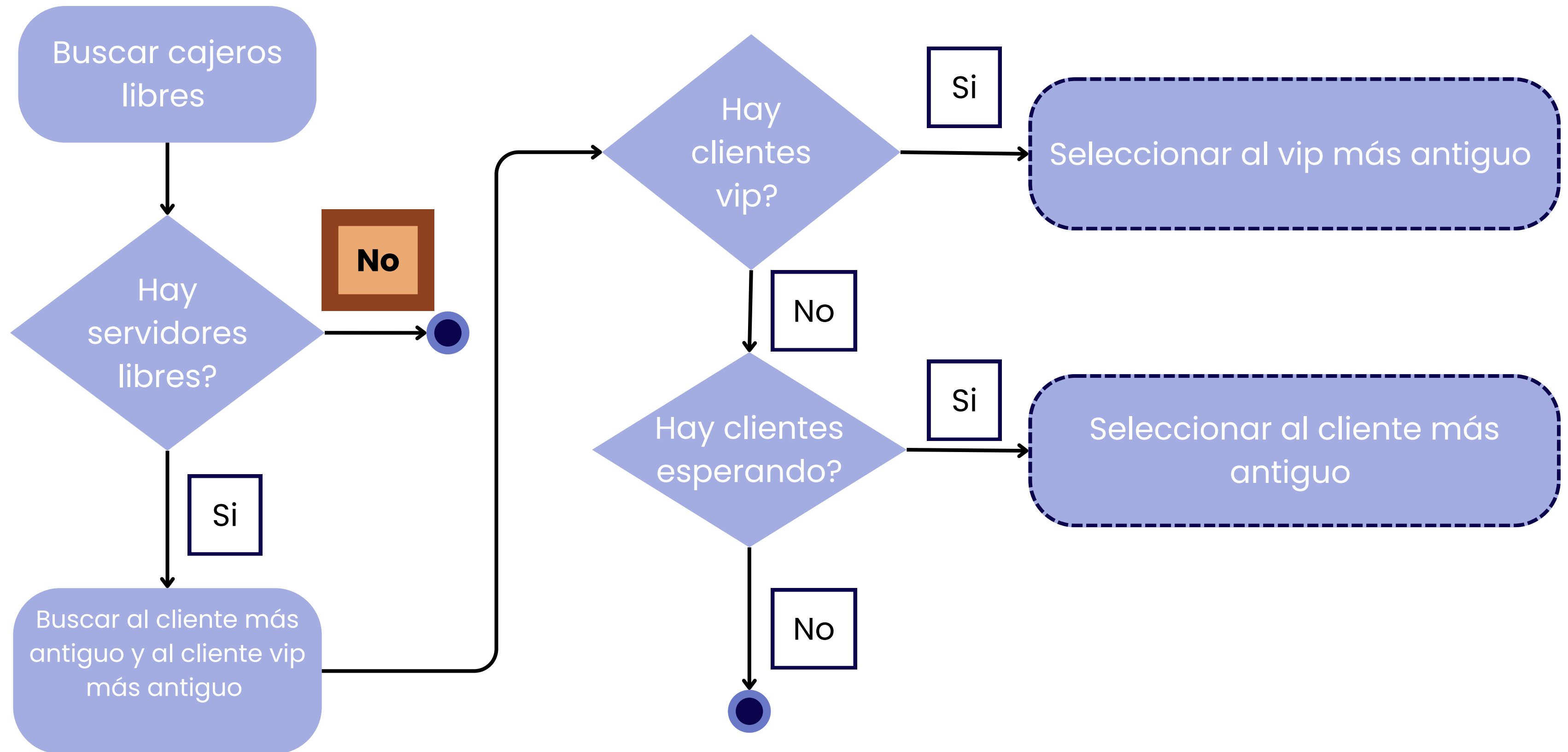
Estadístico F: 84.0878305798584

Valor p: 6.238212291324311e-24

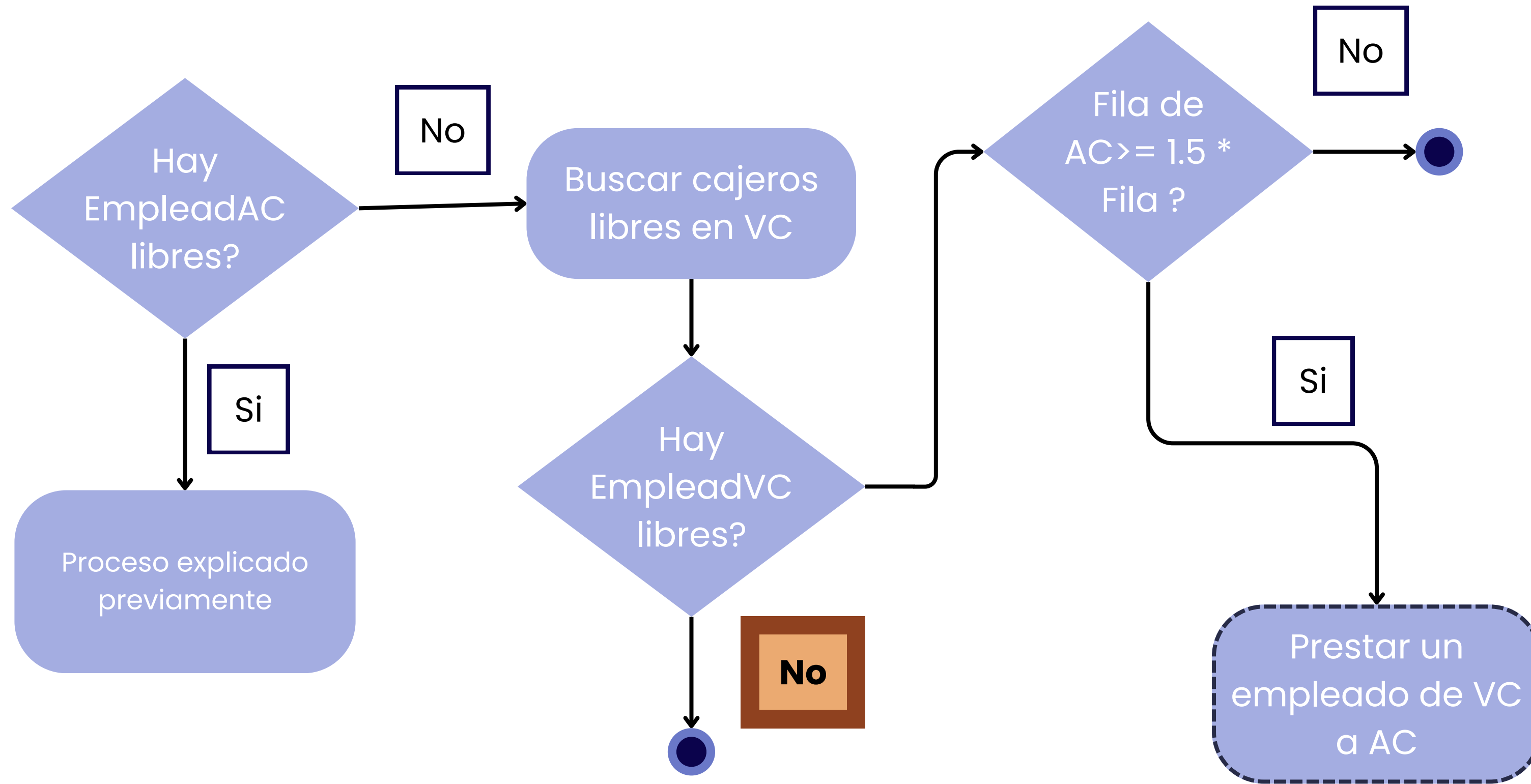
Hay diferencias significativas entre los valores de funcionObjetivo de los simuladores.

OPTIMIZACIÓN

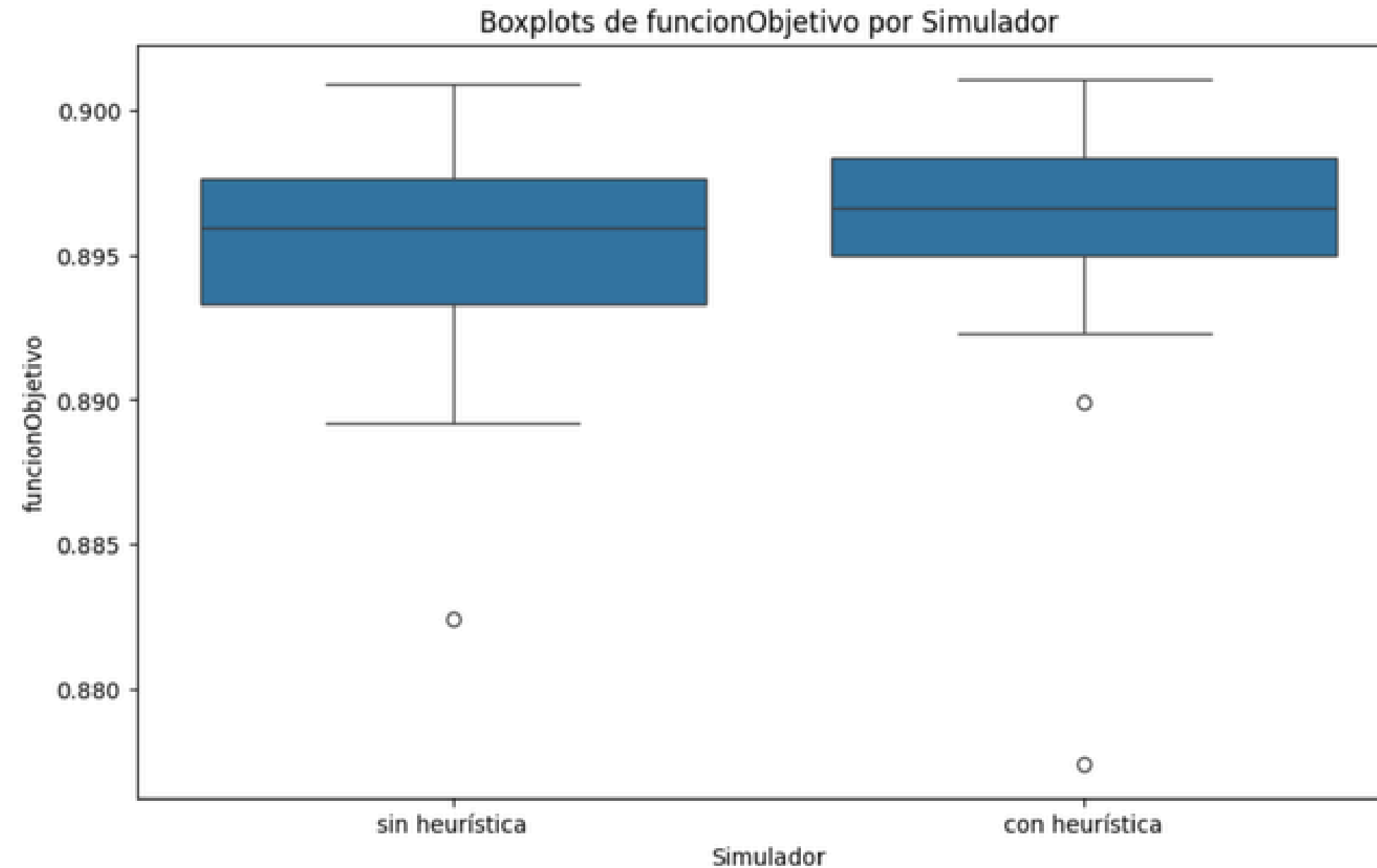
# Función para asignar clientes (se replica para todos los servicios)



# Heurística para prestar empleados (se implementa en VC y AC)



# Resultados de la Heurística para prestar empleados



Resultado del test de ANOVA:

F-value: 1.3395792704629879

P-value: 0.2499213719622442

El P-value es 0.2499, que es mayor que 0.05. Por lo tanto, no podemos rechazar la hipótesis nula.

Esto sugiere que no hay una diferencia significativa en los valores de funcionObjetivo entre los distintos valores de Simulador.

# CONCLUSIÓN

# Recomendación

La mejor solución para el banco será la presentada en la **experimentación 4**.

4 empleados  
Atención Comercial

2 empleados  
Ventanilla de Caja

2 capacitaciones  
mensuales

9 mantenimientos anuales  
Cajeros Automáticos





## 1er cuatrimestre – 2024

- Entrega Final
- 11.63 – Simulación



## Integrantes

- Magdalena Eppens
- Sofía Gonzalez del Solar
- Nicole Reiman
- Francisca Sulzberger

ANEXO

# Mantenimiento

Properties ×

**transition5 - Transition**

Name:  ☐ Show name ☐ Ignore

Triggered by:

Rate:

Action:

Guard:

Properties ×

**tiempoMantenimiento - Parameter**

Name:  ☒ Show name

Visible: ☒ yes

Type:

Default value:

Properties ×

**transition6 - Transition**

Name:  ☐ Show name ☐ Ignore

Triggered by:

Timeout:

Action:

Guard:

```

// Iterar sobre los empleadosVC para encontrar uno con varLibre
int indiceVC = -1;
for (int i = 0; i < empleadosVC.size(); i++) {
    if (empleadosVC.get(i).varLibre) {
        indiceVC = i;
        break; // Salir del bucle una vez encontrado el empleado libre
    }
}

// Si no se encontró un empleado libre en empleadosVC, evaluar empleadosAC
if (indiceVC == -1) {
    int empleadosLibresAC = 0;
    int indiceEmpleadoLibreAC = -1;

    // Contar la cantidad de empleados libres en empleadosAC
    for (int i = 0; i < empleadosAC.size(); i++) {
        if (empleadosAC.get(i).varLibre) {
            empleadosLibresAC++;
        }
        indiceEmpleadoLibreAC = i; // Guardar el índice del empleado libre encontrado
    }

    // Evaluar si es conveniente pasar un empleado de empleadosAC a empleadosVC
    if (empleadosLibresAC > 1 && indiceEmpleadoLibreAC != -1) {
        int tamañoFilaAC = esperaAC.size();
        int tamañoFilaVC = esperaVC.size();

        // Verificar si la fila de esperaAC es como mínimo un 50% más pequeña que la fila de esperaVC
        if (tamañoFilaAC <= (tamañoFilaVC / 1.5)) {
            // Crear un nuevo empleado en empleadosVC con el nivelEmpleado del empleado en empleadosAC
            EmpleadoAC empleadoACLibre = empleadosAC.get(indiceEmpleadoLibreAC);
            EmpleadoVC empleadoVC= new EmpleadoVC();
            empleadoVC.nivelEmpleado= empleadoACLibre.nivelEmpleado;
            empleadoVC.varLibre = true;
            empleadoVC.goToPopulation(empleadosVC);
            // Imprimir el movimiento de empleados
            System.out.println("Moviendo empleado de AC a VC: " + empleadoACLibre);

            // Incrementar el contador de movimientos
            movimientosACaVC++;

            // Eliminar el empleado de empleadosAC
            remove_empleadosAC(empleadoACLibre);

            indiceVC = empleadosVC.size() - 1; // El nuevo empleado transferido es el último en la lista de empleadosVC
        }
    }
}

```

```

// Si se encontró un empleado libre
if (indiceVC != -1) {
    Cliente clienteSeleccionado = null;
    Cliente clienteVipMasAntiguo = null;
    Cliente clienteMasAntiguo = null;
    double maxTiempoEsperaVip = -1;
    double maxTiempoEsperaGeneral = -1;

    // Iterar sobre los clientes en esperaVC para encontrar el más antiguo y el VIP más antiguo
    for (int j = 0; j < esperaVC.size(); j++) {
        Cliente cliente = esperaVC.get(j);
        // Obtener el cliente en la posición j
        System.out.println("InicioEspera: " + cliente.inicioEspera);
        double tiempoEspera = time() - cliente.inicioEspera;
        System.out.println("El tiempo es: " + tiempoEspera);

        if (tiempoEspera > maxTiempoEsperaGeneral && tiempoEspera < cliente.Tolerancia) {
            clienteMasAntiguo = cliente;
            maxTiempoEsperaGeneral = tiempoEspera;
        }

        if ("vip".equals(cliente.tipoCliente) && tiempoEspera > maxTiempoEsperaVip && tiempoEspera < cliente.Tolerancia) {
            clienteVipMasAntiguo = cliente;
            maxTiempoEsperaVip = tiempoEspera;
        }

        if (tiempoEspera > cliente.Tolerancia) {
            cliente.meFui = true;
            esperaVC.free(cliente);
        }
    }

    // Seleccionar el cliente VIP más antiguo si existe, si no, el cliente más antiguo
    if (clienteVipMasAntiguo != null) {
        clienteSeleccionado = clienteVipMasAntiguo;
    } else {
        clienteSeleccionado = clienteMasAntiguo;
    }

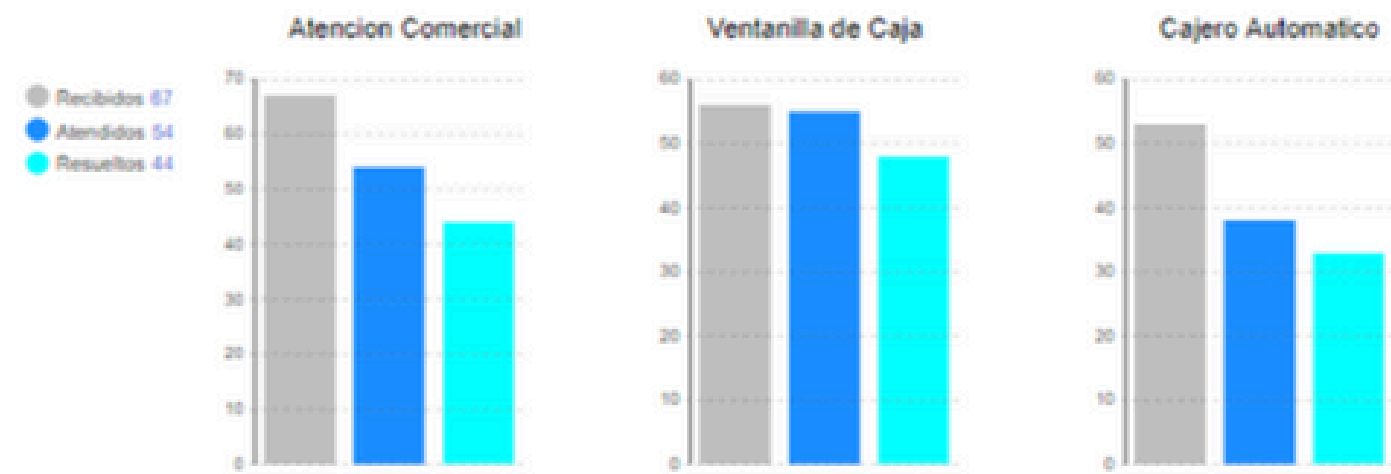
    // Liberar al cliente seleccionado si existe
    if (clienteSeleccionado != null) {
        esperaVC.free(clienteSeleccionado);
        empleadosVC.get(indiceVC).varLibre = false;
        double tiempoEspera = time() - clienteSeleccionado.inicioEspera;
        System.out.println("Cliente procesado: " + clienteSeleccionado);
        System.out.println("Tipo de cliente: " + clienteSeleccionado.tipoCliente);
        System.out.println("Tiempo de espera: " + tiempoEspera);

        // Asignar el cliente seleccionado al empleado libre
        empleadosVC.get(indiceVC).clienteActual = clienteSeleccionado;
        clienteSeleccionado.indice = indiceVC;
    }
}

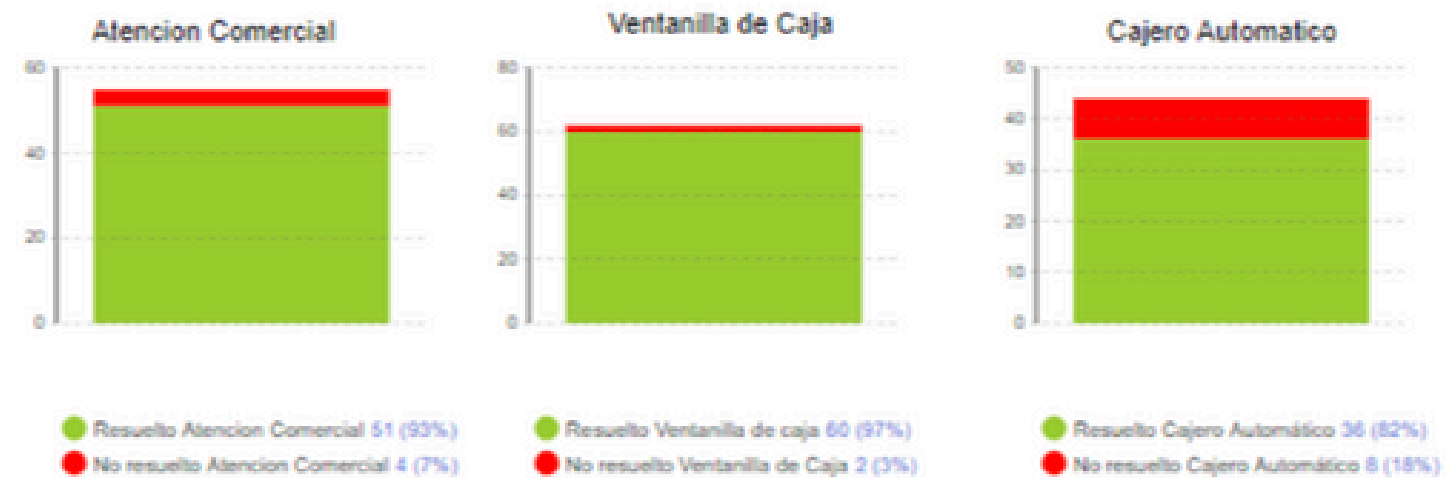
```

# Verificación

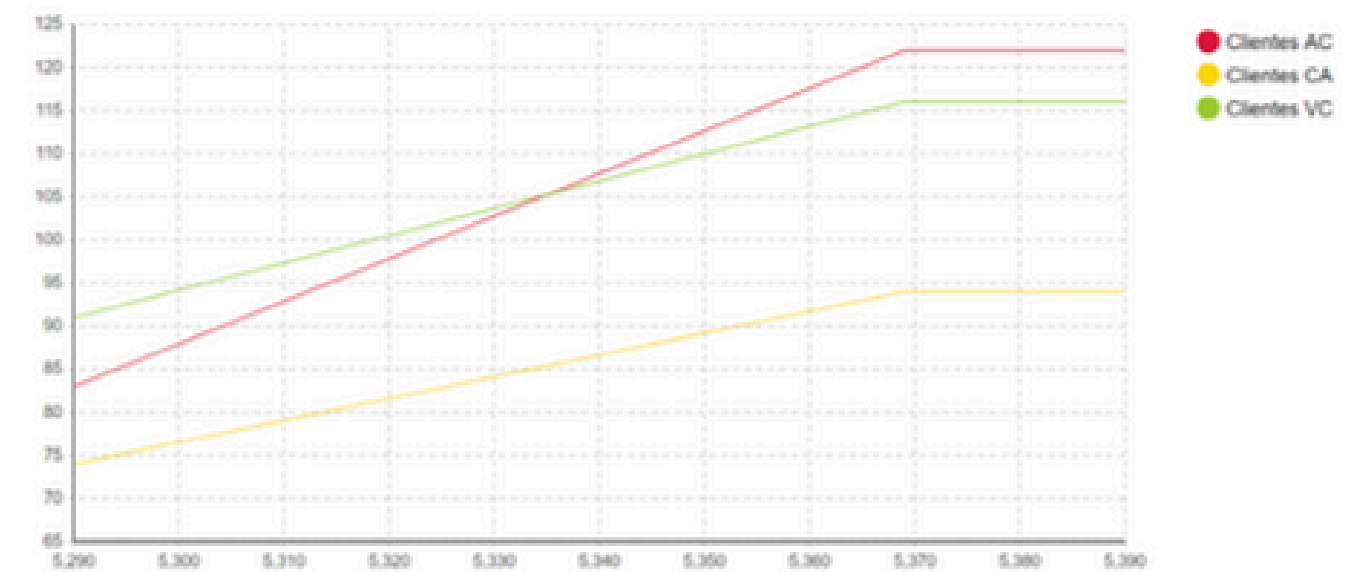
OPERACIONES RECIBIDAS - ATENDIDAS - RESULETAS



CONSULTAS RESUELTAS POR OPERACION



CLIENTES RECIBIDOS

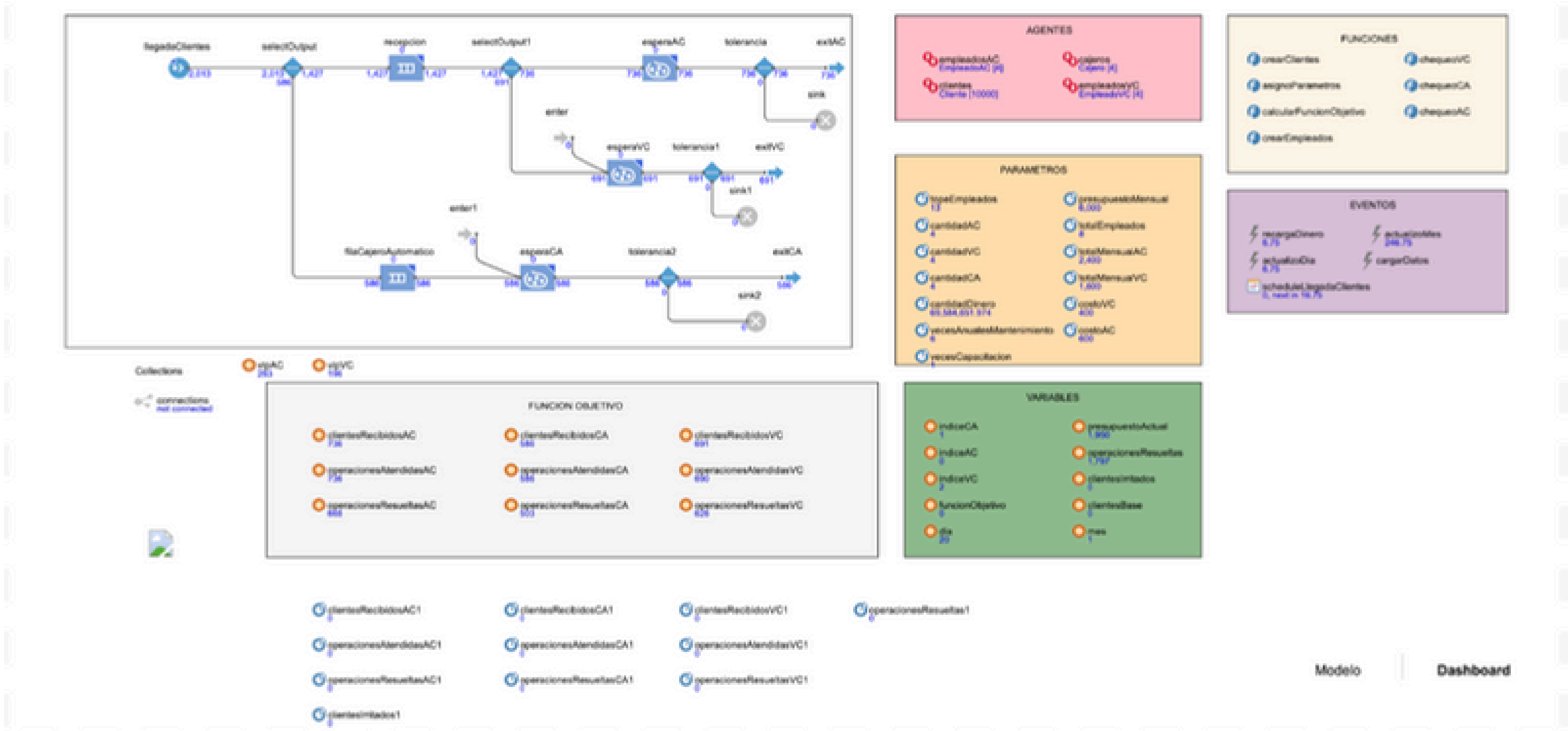


# Verificación

Problems		
2 error(s)		
	Description	Location
✖	The method add(EmpleadoAC) is undefined for the type Ma...	tp_Final_Simulaci...
✖	The method remove(int) is undefined for the type Main._em...	tp_Final_Simulaci...

```
Cliente procesado: 14352
Tipo de cliente: normal
Tiempo de espera: 0.0
root enviado a 0
Cliente procesado: 14349
Tipo de cliente: normal
Tiempo de espera: 10.319535072543658
root enviado a 3
Cliente procesado: 14353
Tipo de cliente: vip
Tiempo de espera: 5.6291880491771735
root enviado a 1
InicioEspera: 199445.93539761796
El tiempo es: 0.0
```

# Validación



# Validación

Servicio	Porcentaje de los clientes totales inicial datos	Porcentaje de los clientes totales simulación
Ventanilla de Caja (VC)	34%	34%
Atención Comercial (AC)	29%	29%
Cajero Automático (CA)	37%	36%

Tipo de cliente	Probabilidad Inicial Datos	Probabilidad Sim
normal	71.9%	71.6%
vip	28.1%	28.4%

Tipo de cliente	Probabilidad Inicial Datos	Probabilidad Simulación
normal	63%	64%
vip	37%	36%