FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Estudo do funcionamento de uma fábrica de skates

SCC

Carlos Lima – 2017266922

Diogo Paula - 2017267117

Arquitetura do Simulador

O objetivo deste trabalho é descrever um cenário onde um empreendedor está a planear construir uma fábrica de skates, mas pretende avaliar a capacidade da fábrica e o investimento necessário. Esta fábrica apenas fabrica pranchas e rodas. Para além de construir skates, a fábrica também irá vender conjuntos de pranchas e rodas em separado de 8 e 4 unidades cada, respetivamente.

Descrição da abordagem

Para a realização deste trabalho, criámos 5 classes principais.

A primeira classe denominada Prancha(), vai conter todos os passos requeridos para o fabrico de um lote de pranchas de um skate, sendo no fim, colocado num armazém de nome storage2

A segunda classe denominada Roda(), vai conter todas as etapas requeridas para o fabrico de um lote de rodas de um skate, sendo no fim colocado num armazém de nome storage4.

A terceira classe denominada Finish(), vai ser utilizada para a montagem do skate em concreto e para construir os conjuntos de pranchas e rodas que serão vendidos à parte. Para este objetivo criámos uma variável que pode assumir três valores (“Packing” ,“Assembly” ou “PackingWheel”) e consoante o valor dela, ou montávamos um skate (“Assembly”) ou construíamos o conjunto de pranchas (“Packing) ou rodas (“PackingWheel”). Para a geração deste valor aleatório utilizámos a função random implementada já nas librarias do Python.

A quarta classe denominada Dia(), vai ser utilizada para simular o horário de funcionamento da fábrica, esta classe, vai alterar o estado de uma variável de “aberto” para “fechado” ou vice-versa consoante o tempo que passa.

Por fim, a última classe denominada Gerador() é a classe que inicia a produção de todos os produtos no horário de funcionamento estabelecido.

Código – Classes

```
class Prancha(sim.Component):
    def process(self):
        global n_placas
        self.enter(pressqueue) #Entrar na Queue do Pressing
        yield self.request((prensa, 4)) #Pedir as prensas necessárias
        self.leave(pressqueue) #Sair do Queue do Pressing
        yield self.hold(100) #Tempo do Pressing
        self.release((prensa, 4)) #Libertar as prensas
        self.enter(storage1queue) #Após terminar o pressing, meter no armazem

        yield self.hold(24*60) #Ficar no storage1 1 dia para dps continuar

        storage1queue.pop() #Retirar do storage1
        self.enter(cutqueue) # Entrar na queue no cutting
        yield self.request((worker, 3)) #Pedir os workers
        self.leave(cutqueue) #Sair da Queue do Cutting
        yield self.hold(60) #Tempo do Cutting
        self.release((worker, 3)) #Libertar os workers


        self.enter(finqueue) #Entrar na Fin Queue
        yield self.request((worker, 1)) #Pedir 1 worker
        self.leave(finqueue) #Sair do Queue
        yield self.hold(15) #Tempo do Finishing
        self.release((worker, 1)) #Libertar o Worker


        self.enter(paintqueue) #Entrar Paint Queue
        yield self.request((worker, 1)) #Pedir 1 worker
        self.leave(paintqueue) #Sair da Paint Queue
        yield self.hold(20) #Tempo do Painting
        self.release((worker, 1)) #Libertar worker

        self.enter(storage2queue)#Após estar pronta entra no storage2
        yield self.hold(24*60) #Esperar na area de secagem 1 dia
        n_placas+=24 #Um lote de placas criados
        prancha.set_capacity(n_placas) #Aumentar o numero de resources disponiveis
```

Nesta classe começamos por colocar a prancha na Queue do pressing, pedimos os recursos necessários, neste caso 4 prensas, tiramos a prancha da Queue para começar a trabalhar nela, esperamos o tempo necessário da etapa em questão e no fim libertamos os recursos. Após a primeira etapa (Pressing) colocamos a placa no armazem durante 24h. Após isto, repetimos o mesmo para todas as etapas, e no fim colocamos a prancha no armazem2. Atualizamos a quantidade do recurso disponível.

1. Classe Prancha

```
class Roda(sim.Component):
    def process(self):
        global n_rodas

        self.enter(foundryqueue)
        yield self.request((fornalha, 1))
        self.leave(foundryqueue)
        yield self.hold(55)
        self.release((fornalha, 1))

        self.enter(storage3queue)
        yield self.hold(24*60)

        storage3queue.pop()
        self.enter(machiningqueue)
        yield self.request((torno, 2))
        self.leave(machiningqueue)
        yield self.hold(60)
        self.release((torno,2))


        self.enter(printingqueue)
        yield self.request((impressora, 1))
        self.leave(printingqueue)
        yield self.hold(20)
        self.release((impressora, 1))


        self.enter(storage4queue)
        yield self.hold(24*60)
        n_rodas += 192
        roda.set_capacity(n_rodas)
```

Nesta classe Roda, fazemos o mesmo que fazemos na classe Prancha, contudo no fim colocamos no armazém 4

2. Classe Roda

```
class Finish(sim.Component):
    def process(self):
        global Conjunto_placa, Conjunto_Roda, n_skate, n_rodas, n_placas
        escolha = random.choice(finish)
        if(escolha == "packing"):
            self.enter(packpranchaqueue)
            yield self.request((worker,2), (prancha, 8*12))
            self.leave(packpranchaqueue)

            for _ in range(8*12):
                storage2queue.pop()
            yield self.hold(10)
            self.release((worker, 2))
            Conjunto_placa += 12
        elif(escolha == "packingwheel"):
            self.enter(packwheelqueue)
            yield self.request((maq_embalagem, 1), (roda, 4*48))
            self.leave(packwheelqueue)
            for _ in range(4*48):
                storage4queue.pop()
            yield self.hold(30)
            self.release((maq_embalagem, 1))
            Conjunto_Roda += 48
        elif(escolha == "assembly"):
            self.enter(assemblyqueue)
            yield self.request((worker, 2), (prancha, 24), (roda, 24*4))
            self.leave(assemblyqueue)
            for _ in range(24):
                storage2queue.pop()
            for _ in range(24*4):
                storage4queue.pop()
            yield self.hold(30)
            self.release((worker, 2))
            n_skate += 24
```

Esta classe, tem como objetivo, ou criar os pacotes de rodas ou de pranchas para vender à parte, ou juntar as rodas e pranchas e fazer os skates.

Para este objetivo, como dito anteriormente, utilizámos uma variável que pode tomar o valor de “Assembly”, “Packing” ou “PackingWheel”. Se for Packing, pedimos 8*12 pranchas e mais 2 workers e tambem retiramos essas 8*12 pranchas do armazém, pois cada lote tem 12 pranchas e cada conjunto de pranchas vão ser 8. Aguardamos o tempo de Packing (self.hold(10)) e no fim libertamos os workers e incrementamos o número de conjuntos feitos.

Isto repete-se para PackingWheel e Assembly. Mudando apenas os recursos necessários e a quantidade.

3. Classe Finish

```
class Dia(sim.Component):
    def process(self):
        global horario
        while True:
            horario = "aberta"
            yield self.hold(60*8)
            horario = "fechado"
            yield self.hold(16*60)

#Generator
class Gerador(sim.Component):
    def process(self):
        global horario
        while (horario == "aberta"):




            for _ in range(n_placas_por_dia):
                Prancha()
                yield self.hold(sim.Uniform(5, 15).sample())
            for _ in range(n_rodas_por_dia):
                Roda()
                yield self.hold(sim.Uniform(5,15).sample())
            for _ in range(n_produtos_por_dia):
                Finish()
                yield self.hold(sim.Uniform(5, 15).sample())
```

A classe Dia, serve para gerir o horário de funcionamento da fábrica. Metendo uma variavel “aberta” ou “fechada”. Depois na Classe Gerador, se a fábrica estiver aberta começamos a produzir as pranchas, rodas e empacotar os produtos.

4. Classes Dia e Gerador.

Resultados Finais

Os resultados finais em relação ao número de skates, conjunto de placas e conjunto de rodas que obtivemos foram os seguintes:

```
Skates: 5280
Caixas de Rodas: 3648
Caixa de placas: 1008
```

Estes resultados são influenciados pela variavel aleatória previamente falada neste relatório. Pois tendo uma maior probabilidade de sair a palavra “Assembly” vai fazer com que haja mais skates feitos e menos conjuntos de peças. Vice-versa também ocorre. Ou seja, a variável aleatória vai influenciar muito o resultado final de produção.

Para além dessa variável aleatória, a quantidade de recursos disponíveis na fábrica vai influenciar as filas de espera. Quanto menos recursos tivermos disponíveis, mais tempo os produtos estarão na fila de espera. Ao termos muitos recursos apesar das filas de espera serem praticamente inexistentes, visto que, como há sempre recursos disponíveis não é necessário haver um tempo de espera, o custo financeiro iria ser muito grande. O que não compensaria o lucro obtido.

Dando como exemplo a etapa Foundry do fabrico das rodas que necessita de uma fornalha.

Se na fábrica inteira apenas tivermos 1 fornalha:

```
fornalha = sim.Resource("fornalha", capacity=1)
```

A queue da etapa Foundry vai ser:

```
Statistics of Foundry Queue at     34560
                                                                    all    excl.zero         zero
------------------------------------------ -------------- ------------ ------------ ------------
Length of Foundry Queue                    duration          34560        34560           0
                                           mean                728.215      728.215
                                           std.deviation       405.024      405.024

                                           minimum               0            0
                                           median              921.423      921.423
                                           90% percentile     1128         1128
                                           95% percentile     1154         1154
                                           maximum            1180         1180

Length of stay in Foundry Queue            entries             556          555           1
                                           mean              12510.810    12533.352
                                           std.deviation      7219.755     7206.682

                                           minimum               0           40.529
                                           median            12529.612    12550.461
                                           90% percentile    22475.419    22480.182
                                           95% percentile    23712.726    23714.910
                                           maximum           24953.231    24953.231
```

Contudo, se tivermos por exemplo 500 fornalhas:

```
fornalha = sim.Resource("fornalha", capacity=500)
```

A queue da etapa Foundry já vai estar:

```
Statistics of Foundry Queue at     34560
                                                                    all    excl.zero         zero
------------------------------------------ -------------- ------------ ------------ ------------
Length of Foundry Queue                    duration          34560        34560            0
                                           mean                  0            0
                                           std.deviation         0            0

                                           minimum               0            0
                                           median                0            0
                                           90% percentile        0            0
                                           95% percentile        0            0
                                           maximum               0            0

Length of stay in Foundry Queue            entries            1440            0         1440
                                           mean                  0
                                           std.deviation         0

                                           minimum               0
                                           median                0
                                           90% percentile        0
                                           95% percentile        0
                                           maximum               0
```

Ou seja, como podemos observar o tempo em que as rodas ficam nesta etapa é 0, porque há sempre fornalhas disponíveis para utilizar, logo, as rodas entram na queue e saem logo por termos recursos disponíveis.

Utilizando valores mais realistas, por exemplo, 5 fornalhas, podemos observar:

```
------------------------------------------ -------------- ------------ ------------ ------------
Length of Foundry Queue                    duration          34560        34560            0
                                           mean                 31.151       31.151
                                           std.deviation        42.977       42.977

                                           minimum               0            0
                                           median                0.896        0.896
                                           90% percentile      106          106
                                           95% percentile      124          124
                                           maximum             141          141

Length of stay in Foundry Queue            entries            1440         1434            6
                                           mean                747.631      750.759
                                           std.deviation       429.920      428.084

                                           minimum               0            1.671
                                           median              718.181      718.929
                                           90% percentile     1376.601     1377.057
                                           95% percentile     1430.016     1430.413
                                           maximum            1542.424     1542.424
```

Ou seja, comparando com o uso de 1 fornalha, o tempo melhorou significativamente.

Estes tipos de melhorias acontecem em todas as queues (exceto os armazéns, pois estes sempre terão produtos lá dentro).

Concluindo, é necessário estabelecer um número de recursos para diminuir o mais possível os tempos de espera nas queues, mas que esteja dentro do orçamento para evitar prejuízos na fábrica.