DJPROD (Casos de Prueba)

"Orquestando la producción"

Materia: 75.39 Aplicaciones Informáticas

Alumno: Juan Sebastián Goldberg

Padrón: 82078

Mail: sebas.goldber@gmail.com

Fecha: 17-NOV-2014

Contenido

Casos de Prueba

En las siguientes secciones se expondrán dos de los casos de prueba implementados:

- En el primero se realiza una prueba planificando la producción de stands.
- En el segundo se realiza una prueba planificando la producción de neumáticos.

Ambos casos de prueba extienden un caso de prueba base que es el que implementa la validación de cronogramas planificados. Los ítems que se validan son los siguientes:

- **Cantidad planificada**: Se verifica que la cantidad planificada de cada tarea se corresponda con la cantidad necesaria para llevar a cabo la producción de todos los productos del pedido planificado.
- **Fechas dentro de calendario**: Se verifica que se respeten las fechas y horarios del calendario de producción. Básicamente que ningún intervalo planificado caiga fuera de los horarios y fechas permitidas.
- **Dependencias**: Se verifican las dependencias. En definitiva es verificar que en ningún instante ocurra que una tarea dependiente supere en cantidad a una tarea de la cual depende.

Caso de Prueba Base

En el caso de prueba base no se prueba nada, simplemente se definen los métodos para verificar que la planificación de un cronograma es correcta.

planificacion/tests/base.py

```
# coding=utf-8
from django.test import TestCase
from produccion.models import *
from planificacion.models import *
from calendario.models import DiaSemana
import datetime
```

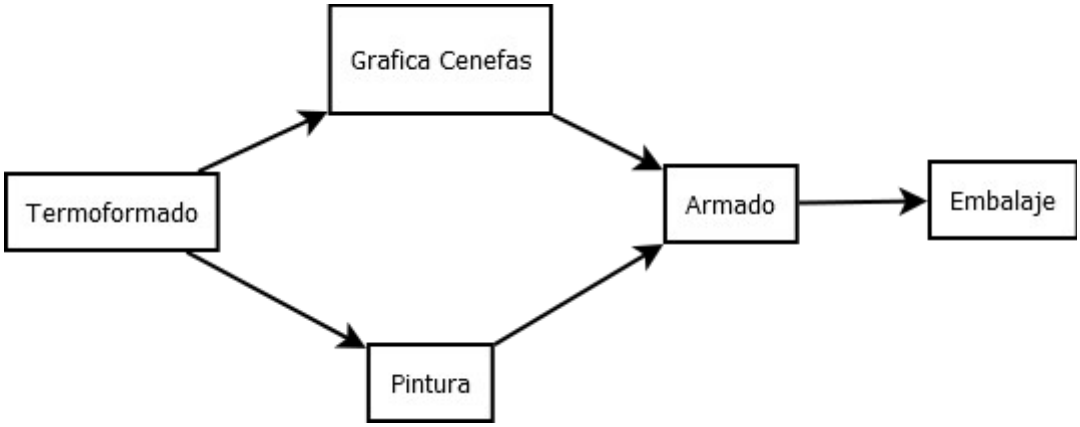
```
import pytz
from django.utils.translation import ugettext as _
from datetime import time as T
from datetime import datetime as DT
from decimal import Decimal as D
from calendario.models import *
from django.conf import settings
import os

utc=pytz.UTC



class PlanificadorTestCase(TestCase):

    def verificar_cantidad_planificada(self, cronograma):

        # Se verifica que se haya planificado la cantidad
        # que corresponde de cada tarea.
        for pedido in cronograma.get_pedidos():
          for item in pedido.get_items():
            for tarea in item.producto.get_tareas():
              cantidad_tarea = cronograma.intervalocronograma_set.filter(
                tarea=tarea,item=item).aggregate(
                models.Sum('cantidad_tarea'))['cantidad_tarea__sum']
              self.assertLessEqual(abs(item.cantidad - cantidad_tarea),
                cronograma.get_tolerancia(item.cantidad),
                'Intervalos involucrados: %s' % cronograma.intervalocronograma_set.filter(
                  tarea=tarea,item=item))


    def verificar_calendario(self, cronograma):
        """
        Se verifica que los intervalos planificados se encuentren dentro
        del calendario definido.
        """
        for i in cronograma.get_intervalos():
            calendario = i.maquina.get_calendario()
            self.assertTrue(
                    calendario.contiene_hueco_completo(
                        desde=i.fecha_desde, hasta=i.fecha_hasta))



    def verificar_dependencias(self, cronograma):
        """
        Se verifica que se respeten las dependencias entre las tareas.
        Básicamente la cantidad de tarea dependiente no puede superar
        en cantidad a la tarea de la cual depende.
        """
        for i in cronograma.get_intervalos():
```

```
        gerenciador_dependencias = cronograma.get_gerenciador_dependencias(
            i.item)
        try:
            gerenciador_dependencias.verificar_modificar_instante(i)
        except ValidationError:
            self.fail(_(u'Error en dependencias para el intervalo %s') % i)
```

Caso de Prueba de Producción de Stands

En esta prueba lo que se hace es definir un calendario de trabajo comprendido de lunes a viernes de 8 a 12 y de 13 a 17.

Se crea un tipo de producto de Stand, el cual tiene definido el siguiente flujo de producción:



Para realizar las tareas indicadas en el flujo de trabajo tenemos los siguientes recursos (se lista recurso, y por cada uno las tareas que realiza):

1. Termoformadora1:
    a. Termoformado
2. Termoformadora2:
    a. Termoformado
3. Plotter:
    a. Grafica Cenefas
4. Cabina Pintura:
    a. Pintura
5. Mano de Obra (operario 1):
    a. Armado
    b. Embalaje

Por último generamos un pedido con un único ítem de 50 Stands, lo planificamos a través de un cronograma y verificamos que el mismo sea correcto.

planificacion/tests/test_stand.py

```
# coding=utf-8
from .base import *

class StandsTestCase(PlanificadorTestCase):

    def test_planificacion(self):
```

```
"""
Se define calendario lu a vi de 8 a 12 y de 13 a 17.
"""
calendario = CalendarioProduccion.get_instance()

calendario.add_intervalos_laborables(
  dias_laborables=[DiaSemana.LUNES,DiaSemana.MARTES,
    DiaSemana.MIERCOLES,DiaSemana.JUEVES,DiaSemana.VIERNES],
  hora_desde=T(8), hora_hasta=T(12))

calendario.add_intervalos_laborables(
  dias_laborables=[DiaSemana.LUNES,DiaSemana.MARTES,
    DiaSemana.MIERCOLES,DiaSemana.JUEVES,DiaSemana.VIERNES],
  hora_desde=T(13), hora_hasta=T(17))

"""
Se define el, producto, las tareas que lo componene y los recursos
a utilizar.
"""
producto = Producto.objects.create(descripcion='Stand')

tarea_armado = Tarea.objects.create(
    descripcion='Armado', tiempo=18)
tarea_embalaje = Tarea.objects.create(
    descripcion='Embalaje', tiempo=9)
tarea_grafica_cenefas = Tarea.objects.create(
    descripcion='Grafica Cenefas', tiempo=12)
tarea_pintura = Tarea.objects.create(
    descripcion='Pintura', tiempo=24)
tarea_termoformado = Tarea.objects.create(
    descripcion='Termoformado', tiempo=30)

maquina_mano_obra = Maquina.objects.create(descripcion='Mano de Obra')
maquina_plotter = Maquina.objects.create(descripcion='Plotter')
maquina_cabina_pintura = Maquina.objects.create(descripcion='Cabina Pintura')
maquina_termoformadora_1 = Maquina.objects.create(descripcion='Termoformadora 1')
maquina_termoformadora_2 = Maquina.objects.create(descripcion='Termoformadora 2')

"""
Se indica qué recurso realiza qué tarea(s).
"""
maquina_mano_obra.add_tarea(tarea_armado)
maquina_mano_obra.add_tarea(tarea_embalaje)
maquina_plotter.add_tarea(tarea_grafica_cenefas)
maquina_cabina_pintura.add_tarea(tarea_pintura)
maquina_termoformadora_1.add_tarea(tarea_termoformado)
maquina_termoformadora_2.add_tarea(tarea_termoformado)

"""
Se indica de que tareas se compone el producto a realizar.
```

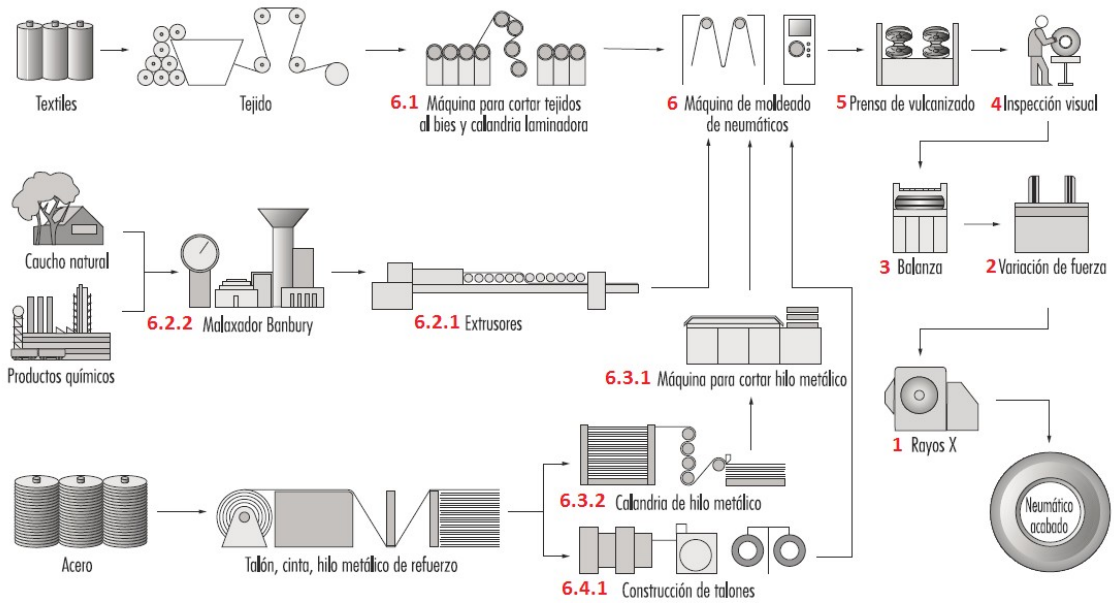
```
"""
producto.add_tarea(tarea_armado)
producto.add_tarea(tarea_embalaje)
producto.add_tarea(tarea_grafica_cenefas)
producto.add_tarea(tarea_pintura)
producto.add_tarea(tarea_termoformado)

"""
Se define el flujo de trabajo a través de las dependencias:
termoformado ---> grafica_cenefas --> armado ---> embalaje
          ---> pintura        -->
"""
producto.add_dependencia_tareas(
   tarea_anterior=tarea_grafica_cenefas, tarea=tarea_armado)
producto.add_dependencia_tareas(
   tarea_anterior=tarea_pintura, tarea=tarea_armado)
producto.add_dependencia_tareas(
   tarea_anterior=tarea_armado, tarea=tarea_embalaje)
producto.add_dependencia_tareas(
   tarea_anterior=tarea_termoformado, tarea=tarea_grafica_cenefas)
producto.add_dependencia_tareas(
   tarea_anterior=tarea_termoformado, tarea=tarea_pintura)

"""
Se crea un pedido de 50 unidades del producto, se asocia un cronograma
a dicho pedido y se realiza la planificación del mismo a partir del
4 de diciembre del 2014 a las 8 de la mañana.
"""
pedido = PedidoPlanificable.objects.create()
pedido.add_item(producto,50)
cronograma = pedido.crear_cronograma(fecha_inicio=
      TZ.make_aware(DT(2014,12,4,8),
         TZ.get_default_timezone()),
      _particionar_pedidos=False)
cronograma.planificar()

"""
Se verifica que se haya planificado correctamente las cantidades de
tarea en función de la cantidad de unidades de producto a producir.
"""
self.verificar_cantidad_planificada(cronograma)

"""
Se verifica que los intervalos planificados se encuentren dentro
del calendario definido.
"""
self.verificar_calendario(cronograma)

"""
Se verifica que se respeten las dependencias entre las tareas.
Básicamente la cantidad de tarea dependiente no puede superar
```

```
        en cantidad a la tarea de la cual depende.
        """
        self.verificar_dependencias(cronograma)
```

Caso de Prueba de Producción de Neumáticos

En esta prueba lo que se hace es definir un calendario de trabajo en el cual los períodos laborales están comprendidos de lunes a viernes de 8 a 12 y de 13 a 17.

Se crea un tipo de producto de Neumático, el cual tiene definido el siguiente flujo de producción (la tarea 1 del gráfico es la más dependiente):



Para realizar las tareas indicadas en el flujo de trabajo tenemos los siguientes recursos (se lista recurso y por cada uno las tareas que realiza):

- 1 - Rayos X
  - 1 - Verificación final
- 2 - Variador de Fuerza
  - 2 - Pruebas de fuerza
- 3 – Balanza
  - 3 - Pesaje
- 4 - Soprte Para Inspección
  - 4 - Inspección visual
- 5 - Prensa de Vulcanizado
  - 5 - Prensado
- 6 - Moldeadora de Neumáticos 1
  - 6 - Moldeado
- 6 - Moldeadora de Neumáticos 2
  - 6 - Moldeado
- 6.1 - Cortadora Laminadora
  - 6.1.1 - Cortado Tejido
  - 6.1.2 - Laminado Tejido
- 6.2.1 - Extructora de Caucho

    - 6.2.1 - Extrucción Caucho
- 6.2.2 - Malaxador Banbury
    - 6.2.2 - Obtención Caucho
- 6.3.1 - Cortadora de Hilo Metálico
    - 6.3.1 - Cortado Hilo Metálico
- 6.3.2 - Calandria Laminadora de Hilo Metálico
    - 6.3.2 - Laminado Hilo Metálico
- 6.4.1 - Constructora de Talones
    - 6.4.1 - Construcción de Talones

Por último generamos un pedido con un único ítem con 200 Neumáticos, lo planificamos a través de un cronograma y verificamos que el mismo sea correcto.

planificacion/tests/test_neumatico.py

```
# coding=utf-8
from .base import *

class NeumaticosTestCase(PlanificadorTestCase):

    def test_planificacion(self):

        """
        Se define calendario lu a vi de 8 a 12 y de 13 a 17.
        """
        calendario = CalendarioProduccion.get_instance()

        calendario.add_intervalos_laborables(
          dias_laborables=[DiaSemana.LUNES,DiaSemana.MARTES,
            DiaSemana.MIERCOLES,DiaSemana.JUEVES,DiaSemana.VIERNES],
          hora_desde=T(8), hora_hasta=T(12))

        calendario.add_intervalos_laborables(
          dias_laborables=[DiaSemana.LUNES,DiaSemana.MARTES,
            DiaSemana.MIERCOLES,DiaSemana.JUEVES,DiaSemana.VIERNES],
          hora_desde=T(13), hora_hasta=T(17))

        """
        Se define el, producto, las tareas que lo componene y los recursos
        a utilizar.
        """
        producto = Producto.objects.create(descripcion='Neumático')

        _1_verificacion = Tarea.objects.create(
            descripcion=u'1 - Verificación final', tiempo=.2)
        _2_pruebas_fuerza = Tarea.objects.create(
            descripcion=u'2 - Pruebas de fuerza', tiempo=2)
        _3_pesaje = Tarea.objects.create(
            descripcion=u'3 - Pesaje', tiempo=.15)
        _4_inspeccion = Tarea.objects.create(
            descripcion=u'4 - Inspección visual', tiempo=.5)
```

```
_5_prensado = Tarea.objects.create(
    descripcion=u'5 - Prensado', tiempo=1.5)
_6_moldeado = Tarea.objects.create(
    descripcion=u'6 - Moldeado', tiempo=2)
_6_1_1_cortado_tejido = Tarea.objects.create(
    descripcion=u'6.1.1 - Cortado Tejido', tiempo=.2)
_6_1_2_laminado_tejido = Tarea.objects.create(
    descripcion=u'6.1.2 - Laminado Tejido', tiempo=.4)
_6_2_1_extruccion_caucho = Tarea.objects.create(
    descripcion=u'6.2.1 - Extrucción Caucho', tiempo=.6)
_6_2_2_obtencion_caucho = Tarea.objects.create(
    descripcion=u'6.2.2 - Obtención Caucho', tiempo=.4)
_6_3_1_cortado_hilo_metalico = Tarea.objects.create(
    descripcion=u'6.3.1 - Cortado Hilo Metálico', tiempo=.1)
_6_3_2_laminado_hilo_metalico = Tarea.objects.create(
    descripcion=u'6.3.2 - Laminado Hilo Metálico', tiempo=.4)
_6_4_1_construccion_talones = Tarea.objects.create(
    descripcion=u'6.4.1 - Construcción de Talones', tiempo=2)

_1_rayos_x = Maquina.objects.create(
      descripcion=u"1 - Rayos X")
_2_variador_de_fuerza = Maquina.objects.create(
      descripcion=u"2 - Variador de Fuerza")
_3_balanza = Maquina.objects.create(
      descripcion=u"3 - Balanza")
_4_soprte_para_inspeccion = Maquina.objects.create(
      descripcion=u"4 - Soprte Para Inspección")
_5_prensa_de_vulcanizado = Maquina.objects.create(
      descripcion=u"5 - Prensa de Vulcanizado")
_6_moldeadora_de_neumaticos_1 = Maquina.objects.create(
      descripcion=u"6 - Moldeadora de Neumáticos 1")
_6_moldeadora_de_neumaticos_2 = Maquina.objects.create(
      descripcion=u"6 - Moldeadora de Neumáticos 2")
_6_1_cortadora_laminadora = Maquina.objects.create(
      descripcion=u"6.1 - Cortadora Laminadora")
_6_2_1_extructora_de_caucho = Maquina.objects.create(
      descripcion=u"6.2.1 - Extructora de Caucho")
_6_2_2_malaxador_banbury = Maquina.objects.create(
      descripcion=u"6.2.2 - Malaxador Banbury")
_6_3_1_cortadora_de_hilo_metalico = Maquina.objects.create(
      descripcion=u"6.3.1 - Cortadora de Hilo Metálico")
_6_3_2_calandria_laminadora_de_hilo_metalico = Maquina.objects.create(
      descripcion=u"6.3.2 - Calandria Laminadora de Hilo Metálico")
_6_4_1_constructora_de_talones = Maquina.objects.create(
      descripcion=u"6.4.1 - Constructora de Talones")

"""
Se indica qué recurso realiza qué tarea(s).
"""
_1_rayos_x.add_tarea(_1_verificacion)
_2_variador_de_fuerza.add_tarea(_2_pruebas_fuerza)
```

```
_3_balanza.add_tarea(_3_pesaje)
_4_soprte_para_inspeccion.add_tarea(_4_inspeccion)
_5_prensa_de_vulcanizado.add_tarea(_5_prensado)
_6_moldeadora_de_neumaticos_1.add_tarea(_6_moldeado)
_6_moldeadora_de_neumaticos_2.add_tarea(_6_moldeado)
_6_1_cortadora_laminadora.add_tarea(_6_1_1_cortado_tejido)
_6_1_cortadora_laminadora.add_tarea(_6_1_2_laminado_tejido)
_6_2_1_extructora_de_caucho.add_tarea(_6_2_1_extruccion_caucho)
_6_2_2_malaxador_banbury.add_tarea(_6_2_2_obtencion_caucho)
_6_3_1_cortadora_de_hilo_metalico.add_tarea(
      _6_3_1_cortado_hilo_metalico)
_6_3_2_calandria_laminadora_de_hilo_metalico.add_tarea(
      _6_3_2_laminado_hilo_metalico)
_6_4_1_constructora_de_talones.add_tarea(
      _6_4_1_construccion_talones)

"""
Se indica de que tareas se compone el producto a realizar.
"""
producto.add_tarea(_1_verificacion)
producto.add_tarea(_2_pruebas_fuerza)
producto.add_tarea(_3_pesaje)
producto.add_tarea(_4_inspeccion)
producto.add_tarea(_5_prensado)
producto.add_tarea(_6_moldeado)
producto.add_tarea(_6_1_1_cortado_tejido)
producto.add_tarea(_6_1_2_laminado_tejido)
producto.add_tarea(_6_2_1_extruccion_caucho)
producto.add_tarea(_6_2_2_obtencion_caucho)
producto.add_tarea(_6_3_1_cortado_hilo_metalico)
producto.add_tarea(_6_3_2_laminado_hilo_metalico)
producto.add_tarea(_6_4_1_construccion_talones)


"""
Se define el flujo de trabajo a través de las dependencias:
"""
producto.add_dependencia_tareas(
   tarea_anterior=_2_pruebas_fuerza, tarea=_1_verificacion)
producto.add_dependencia_tareas(
   tarea_anterior=_3_pesaje, tarea=_2_pruebas_fuerza)
producto.add_dependencia_tareas(
   tarea_anterior=_4_inspeccion, tarea=_3_pesaje)
producto.add_dependencia_tareas(
   tarea_anterior=_5_prensado, tarea=_4_inspeccion)
producto.add_dependencia_tareas(
   tarea_anterior=_6_moldeado, tarea=_5_prensado)
producto.add_dependencia_tareas(
   tarea_anterior=_6_1_1_cortado_tejido, tarea=_6_moldeado)
producto.add_dependencia_tareas(
   tarea_anterior=_6_2_1_extruccion_caucho, tarea=_6_moldeado)
```

```
producto.add_dependencia_tareas(
   tarea_anterior=_6_3_1_cortado_hilo_metalico, tarea=_6_moldeado)
producto.add_dependencia_tareas(
   tarea_anterior=_6_4_1_construccion_talones, tarea=_6_moldeado)
producto.add_dependencia_tareas(
   tarea_anterior=_6_1_2_laminado_tejido, tarea=_6_1_1_cortado_tejido)
producto.add_dependencia_tareas(
   tarea_anterior=_6_2_2_obtencion_caucho, tarea=_6_2_1_extruccion_caucho)
producto.add_dependencia_tareas(
   tarea_anterior=_6_3_2_laminado_hilo_metalico, tarea=_6_3_1_cortado_hilo_metalico)

"""
Se crea un pedido de 200 unidades del producto, se asocia un cronograma
a dicho pedido y se realiza la planificación del mismo a partir del
4 de diciembre del 2014 a las 8 de la mañana.
"""
pedido = PedidoPlanificable.objects.create()
pedido.add_item(producto,200)
cronograma = pedido.crear_cronograma(fecha_inicio=
      TZ.make_aware(DT(2014,12,4,8),
         TZ.get_default_timezone()),
      _particionar_pedidos=False)
cronograma.planificar()

"""
Se verifica que se haya planificado correctamente las cantidades de
tarea en función de la cantidad de unidades de producto a producir.
"""
self.verificar_cantidad_planificada(cronograma)

"""
Se verifica que los intervalos planificados se encuentren dentro
del calendario definido.
"""
self.verificar_calendario(cronograma)

"""
Se verifica que se respeten las dependencias entre las tareas.
Básicamente la cantidad de tarea dependiente no puede superar
en cantidad a la tarea de la cual depende.
"""
self.verificar_dependencias(cronograma)
```