

DJPROD (Informe Final)

"Orquestando la Producción"

MATERIA: 75.39 APLICACIONES INFORMÁTICAS

ALUMNO: JUAN SEBASTIÁN GOLDBERG

PADRÓN: 82078

FECHA: 27-NOV-2014

Contenido

Contenido	2
Objetivo	5
Ejemplo de problema básico	5
Resultado esperado de la resolución	6
Ideal del producto a desarrollar	6
Justificación	7
Metodología	7
Tipo de infraestructura	8
Cronograma tentativo	8
Detalle del Proceso de Planificación a Implementar	9
Modelo Lineal de Tiempo Continuo (Obtención de Intervalos a Planificar)	9
Heurística Basada en Dependencias (Ordenamiento de los Intervalos)	9
Optimización del Resultado	10
Estado del Arte	11
Mercado objetivo	11
Productos existentes	11
Análisis de productos existentes	11
Presea (NeuralSoft)	11
PlannerOne (Quonext)	12
Super VISION (Lemdi)	12
Preactor (SIMLog)	13
Cuadro de Estado del Arte	13
Conclusión	14
Modelo	15
Infraestructura	15
Casos de Uso	15
Diagrama de Casos de Uso	15
Especificación de Casos de Uso	16
ABM Productos	16
ABM Máquinas	17
ABM Tareas	17
ABM Tareas por Producto	17

ABM Tareas por Máquina	17
ABM Dependencias Tareas por Producto	18
Carga de Tiempos de Producción.....	18
Creación, Planificación, Activación, y Finalización de Cronogramas.....	18
Cancelación de Intervalo de Cronograma	19
Carga de pedidos	19
Creación y Planificación de Cronograma.....	19
Carga de Cantidad de Tarea Producida por Intervalo de Cronograma	19
Finalización de Intervalo de Cronograma	20
Paquetes.....	20
Clases.....	21
Paquete de producción	21
Paquete de planificación	21
Paquete de calendario	21
Diagramas de Secuencia.....	22
Planificación de Cronograma	22
Casos de Prueba	24
Caso de Prueba Base	24
planificacion/tests/base.py	24
Caso de Prueba de Producción de Stands.....	25
planificacion/tests/test_stand.py	26
Ejecución del caso	28
Resultado de la ejecución	29
Caso de Prueba de Producción de Neumáticos	30
planificacion/tests/test_neumatico.py	31
Ejecución del caso	35
Resultado de la ejecución	35
Futuras Líneas de Trabajo	37
Usabilidad de usuario.....	37
Implementación de calendario de máquina	37
Incorporación de materia prima al proceso productivo	37
Incorporación de costos de producción.....	37
Bibliografía	38
Referencias.....	38

Anteproyecto

Objetivo

El objetivo del producto a desarrollar es la planificación en forma automática y optimizada de la producción.

Los puntos clave ofrecidos por el producto son:

- Aprovechar al máximo la maquinaria mejorando los tiempos de producción.
- Calcular fechas de entrega de los distintos pedidos que tenga la organización.
- Detección de cuellos de botella en la producción.

La planificación se realizará en forma automática teniendo los siguientes ítems en consideración:

- Maquinarias de la fabrica
- Tareas que realiza cada máquina
- Productos producidos por la fábrica
- Tareas de las que se compone cada producto
- Dependencia entre tareas por producto
- Tiempos que demora realizar cada tarea por máquina, por producto
- Pedidos a producir

Para llevar a cabo dicha planificación se realiza un proceso de optimización que busca minimizar el tiempo utilizado para concretar los pedidos mencionados. Dicha optimización será realizada aplicando resolución simplex a un modelo lineal como primer aproximación y luego se utilizará una heurística en función de las dependencias existentes entre las tareas. En el caso que no existan dependencias, entonces la solución será siempre óptima.

Ejemplo de problema básico

Se quiere producir tres productos PA, PB, PC. La cantidad a producir de cada producto está dada por los pedido D1, D2, D3. A continuación se muestran los productos solicitados por pedido:

	PA	PB	PC
D1	100	100	
D2	50		200
D3		150	

Para producir cada uno de estos productos se deben combinar algunas de las tareas T1, T2, T3, T4, T5. A continuación se indica la duración en minutos en realizar cada tarea para cada producto. La duración de las tareas es independiente de la máquina, pero podría no serlo:

	T1	T2	T3	T4	T5
PA		15		3	4
PB	2			2	3
PC	3		2	3	

Para realizar dichas tareas se encuentran las máquinas M1, M2, M3. A continuación se indica qué tareas puede realizar cada máquina:

	T1	T2	T3	T4	T5
M1	X	X			
M2		X	X		
M3				X	X

Las dependencias existentes entre las tareas son las siguientes:

- T5 depende de T3
- T5 depende de T2
- T2 depende de T1
- T4 es independiente

Resultado esperado de la resolución

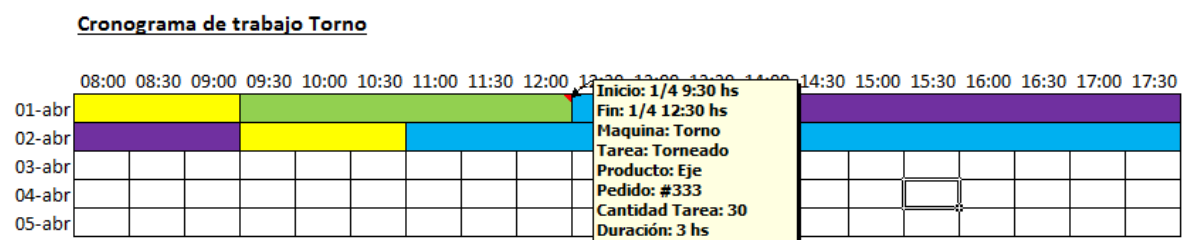
El resultado esperado de la resolución del problema básico será el cronograma en donde estará planificada la realización de las tareas T_i de cada máquina M_j para producir los productos P_k de cada pedido K_l , minimizando el tiempo total de producción.

Ideal del producto a desarrollar

A partir del problema planteado y las restricciones indicadas, el ideal es obtener un cronograma de uso de cada máquina, donde dicho cronograma estará dividido en intervalos de tiempo.

Cada intervalo dependiendo de la tarea que se esté realizando, el producto, y el pedido, será representado con un color distinto. Posicionándose sobre un intervalo, se podrá visualizar el detalle de lo que se estará produciendo en el mismo.

A continuación se muestra un ejemplo del cronograma obtenido para una determinada máquina:



Justificación

El proyecto tiene origen por una necesidad detectada en la que distinto tipo de fábricas no cumplen con las fechas de entrega, en gran medida, por una mala planificación de la producción.

Para entender la justificación del proyecto creo que sería conveniente realizar un breve análisis FODA (Fortalezas, Oportunidades, Debilidades, Amenazas) del producto a desarrollar:

- Fortalezas:
 - Planificación Automática y optimizada de la producción.
 - Producto realizado íntegramente a partir de frameworks y herramientas de software libre.
- Oportunidades:
 - Los productos existentes en el mercado son escasos, muy caros y en general no son de origen nacional.
- Debilidades:
 - Siendo que se trata de una primera etapa, la planificación se verá limitada a la utilización de maquinaria, productos, tareas y pedidos.
- Amenazas:
 - No se han detectado amenazas

Como se comentó en las 'Oportunidades', los productos existentes en el mercado son escasos, muy caros y en general no son de origen nacional.

La planificación de la producción es un factor clave que, realizándola en forma correcta, puede generar valor económico en las organizaciones que se ocupan de llevarla a cabo, obteniendo así, ventaja competitiva en el mercado.

Metodología

La metodología a utilizar será TDD (Test Driven Development, o en español: Desarrollo Guiado por Pruebas).

Está es una metodología ágil que busca definir las pruebas de la aplicación antes que el desarrollo mismo.

Es por esto, que dada la complejidad de la aplicación y el compromiso de la misma para con el futuro cliente, lo principal es asegurar su correcto funcionamiento.

Tipo de infraestructura

La aplicación será WEB y para su implementación se utilizará el framework Django sobre el sistema operativo Ubuntu 14.04 LTS.

Para implementar la resolución del problema utilizando programación lineal mixta, se utilizará las siguientes librerías:

1. PuLP: Para realizar la descripción del modelo en un lenguaje estándar.
2. GLPK: Para realizar la resolución del modelo.

El lenguaje a utilizar para la implementación de la aplicación será Python.

Tener en cuenta que todo el software a utilizar es de uso libre (incluyendo fines comerciales), con lo cual se evita la necesidad de adquisición de licencias y se tiene acceso a todo el código fuente.

Cronograma tentativo

Fecha	Hito
15-sep-2014	Anteproyecto.
22-sep-2014	Ambiente de desarrollo instalado (incluye sistema operativo, frameworks, base de datos y servidor web).
06-oct-2014	Paquetes y clases definidas y codificadas.
20-oct-2014	Interfaz WEB funcionando para realizar la configuración de la aplicación: Maquinas, Tarea, Productos, Tiempos, Dependencias, Pedidos.
03-nov-2014	Pruebas de funcionalidad de planificación sin optimización, definidas y ejecutadas en forma exitosa.
17-nov-2014	Pruebas de funcionalidad de planificación con optimización, definidas y ejecutadas en forma exitosa.
01-dic-2014	Interfaz WEB funcionando para realizar la planificación de pedidos. Primera versión de la aplicación junto con el informe.

Detalle del Proceso de Planificación a Implementar

Modelo Lineal de Tiempo Continuo (Obtención de Intervalos a Planificar)

Se busca asignar en forma óptima, de forma de minimizar el tiempo de producción, el tiempo a invertir de cada máquina, en cada tarea, para cada producto de cada pedido.

La función TMTDP: $f(m,t,p,d) \rightarrow R$, nos devuelve el tiempo a invertir para la máquina m , tarea t , producto p , pedido d .

Si no se tiene en cuenta el ordenamiento de todos los valores posibles de (t,p,d) para una misma máquina, obtener una solución óptima para este problema es algo relativamente sencillo y que se hace en un tiempo casi instantáneo.

La resolución se realizará a través del método SIMPLEX sobre un modelo matemático lineal basado en:

- La cantidad de producto por pedido a planificar.
- Las máquinas utilizadas en la planificación.
- Las tareas que puede realizar cada máquina.
- Las tareas necesarias para fabricar cada producto.
- Los tiempos necesarios para realizar una tarea T en la máquina M para fabricar el producto P .

Heurística Basada en Dependencias (Ordenamiento de los Intervalos)

Luego de obtener los valores para cada punto del dominio de la función TMTDP, podríamos decir que por cada combinación (m,t,p,d) tenemos un bloque de duración $f(m,t,p,d)$.

Estos bloques por supuesto no están ordenados y deberíamos ordenarlos teniendo en cuenta las siguientes restricciones:

- Los bloques de una misma máquina no pueden estar solapados.
- Las tareas dependientes en un producto de un mismo pedido, deben respetar las dependencias. Es decir si T_2 depende de T_1 en el producto PA , entonces la cantidad de tarea T_2 no puede superar a T_1 en ningún instante del cronograma para producir el producto PA de un mismo pedido.

Teniendo estos temas en mente la heurística a aplicar será la siguiente:

1. Para un mismo producto de un mismo pedido:
 - a. Armar un listado ordenado de los bloques según el siguiente criterio: grado de dependencia de las tareas dependientes (las menos dependientes al principio y las más dependientes al final).
 - b. Adicionar los bloques con tareas independientes al final del listado.
 - c. Recorrer el listado de bloques:
 - i. Asignar el bloque a la máquina correspondiente respetando las restricciones mencionadas arriba e intentando asignarlo con la fecha más temprana posible.

Optimización del Resultado

Finalmente una vez ya realizados los 2 pasos anteriores tendremos un cronograma. Dicho cronograma podría mejorarse seguramente en el caso que existan dependencias. Es por esto que para optimizar el resultado obtenido se aplicará un proceso iterativo que intente acercar cada intervalo lo más posible en torno a la fecha de inicio del cronograma.

El proceso iterativo intentará por cada intervalo, en caso que lo anteceda un hueco, cambiar la fecha de inicio a una anterior. La búsqueda de la nueva fecha de inicio se hará en forma binaria sobre el intervalo temporal del hueco validando que se cumplan todas las reglas de negocio definidas (no existencia de solapamientos, que se respeten dependencias entre tareas, etc.).

Estado del Arte

Estado del Arte

Mercado objetivo

Antes que nada es importante establecer el mercado objetivo del producto a desarrollar, ya que de esta forma podremos realizar un mejor análisis de los productos existentes pensando en nuestros posibles clientes.

Este mercado es el de las pequeñas empresas argentinas que se dedican a la fabricación de productos.

Con lo cual hay una serie de items decisivos que deberían evaluarse en los productos existentes:

1. Independencia respecto de otros productos.
2. Complejidad de implementación.
3. Infraestructura necesaria.
4. Precio.
5. Idioma.
6. País de procedencia.

Productos existentes

Si realizamos la siguiente búsqueda en google: “planificacion produccion soft”, tenemos como principales resultados los siguientes:

1. Presea (NeuralSoft): <https://www.neuralsoft.com/productos/presea>
2. PlannerOne (Ortems): <http://www.plannerone.com/?lang=es>
3. Super VISION (Lemdi): <http://www.lemdi.com/es/p5-software.html>
4. Preactor (SIMLog): <http://www.simlog.com.ar/>

La búsqueda la limitamos al idioma español, ya que tomamos como premisa que una pequeña empresa productora argentina, no estaría dispuesta a utilizar una aplicación que no sea en español.

Análisis de productos existentes

Presea (NeuralSoft)

Parecería ser una opción bastante completa en lo que respecta a la planificación de la producción, sin embargo la planificación se realiza en función de un plan de producción, el cual generalmente en las pequeñas empresas está ausente, siendo que se manejan más a nivel de

pedido. También al parecer el producto ofrecido es un sistema integral ERP, con lo cual su complejidad debe ser bastante alta, como así también su precio, y en principio no sería accesible para una pequeña empresa.

1. **Independencia respecto de otros productos:** Independiente
2. **Complejidad de implementación:** Media-Alta
3. **Infraestructura necesaria:** Baja.
4. **Precio:** Medio-Alto.
5. **Idioma Español:** Sí
6. **País de procedencia:** Argentina

PlannerOne (Quonext)

El producto a simple vista parece satisfacer la necesidad de planificar la producción, sin embargo, existen varias desventajas: Es dependiente de la solución Microsoft Dynamics, con lo cual su implementación es bastante compleja y sus costos son altos. Otra desventaja es que el producto es Español, con lo cual el soporte y la localización se pueden ver afectados, como así también los costos.

1. **Independencia respecto de otros productos:** Dependiente de Microsoft Dynamics.
2. **Complejidad de implementación:** Alta.
3. **Infraestructura necesaria:** Alta.
4. **Precio:** Alto.
5. **Idioma Español:** Sí
6. **País de procedencia:** España.

Super VISION (Lemdi)

Existe poca información respecto del producto, y parece bastante desactualizado. La empresa proveedora parece no dedicarse específicamente al desarrollo de software, con lo cual pone en duda el soporte ofrecido y la adaptabilidad y mejora del producto según las necesidades de cada cliente. También parecería ser una aplicación dependiente del sistema operativo y que además no correría en un servidor.

1. **Independencia respecto de otros productos:** Dependiente de sistema operativo.
Aplicación instalable.
2. **Complejidad de implementación:** Poco adaptable.
3. **Infraestructura necesaria:** Media-Baja.
4. **Precio:** Medio-Bajo.
5. **Idioma Español:** Sí

6. **País de procedencia:** Argentina

Preactor (SIMLog)

Parecería ser una opción bastante completa en lo que respecta a la planificación de la producción, sin embargo se verifica que existe dependencia con el sistema operativo y es necesario instalar una aplicación cliente. Los clientes que registra el proveedor, en general son medianas a grandes empresas, con lo cual los procesos de implantación que maneja en cuanto a complejidad y costos deben ser altos.

1. **Independencia respecto de otros productos:** Dependencia de Sistema Operativo. Aplicación instalable.
2. **Complejidad de implementación:** Alta
3. **Infraestructura necesaria:** Alta.
4. **Precio:** Medio-Alto.
5. **Idioma Español:** Sí
6. **País de procedencia:** Argentina

Cuadro de Estado del Arte

A continuación se resume la información mencionada en las secciones anteriores, y se incluye para la aplicación a desarrollar, el objetivo a satisfacer para cada ítem analizado:

Producto	Presea (NeuralSoft)	PlannerOne (Quonext)	Super VISION (Lemdi)	Preactor (SIMLog)	Aplicación a desarrollar
Independencia respecto de otros productos	Independiente	Dependiente de Microsoft Dynamics	Dependiente de sistema operativo. Aplicación instalable	Dependencia de Sistema Operativo. Aplicación instalable.	Independiente
Complejidad de implementación	Media-Alta	Alta	Poco adaptable	Alta	Baja
Infraestructura necesaria	Baja	Alta	Media-Baja	Alta	Baja
Precio	Medio-Alto	Alto	Medio-Bajo	Medio-Alto	Bajo
Idioma Español	Sí	Sí	Sí	Sí	Sí
País de procedencia	Argentina	España	Argentina	Argentina	Argentina
Integración con otros procesos	SI	SI	NO	SI	SI
Cliente Web	SI	NO	NO	NO	SI

Conclusión

Luego del análisis realizado respecto de los principales productos en el mercado, vemos que el que mejor que se adapta a las características buscadas es Presea (NeuralSoft), sin embargo podemos verificar que hay dos características que no se satisfacen:

- **Complejidad de implementación:** Media-Alta
- **Precio:** Medio-Alto.

Estas dos características son fundamentales a la hora de ofrecer un producto a pequeñas empresas, las cuales en general son escasas en recursos.

La idea principal es que la aplicación a desarrollar satisfaga las siguientes características:

1. **Independencia respecto de otros productos:** Independiente.
2. **Complejidad de implementación:** Baja.
3. **Infraestructura necesaria:** Baja.
4. **Precio:** Bajo.
5. **Idioma Español:** Sí
6. **País de procedencia:** Argentina
7. **Integración con otros procesos:** Sí
8. **Cliente Web:** Sí

Por supuesto, satisfacer todas estas características tiene un costo. Dicho costo es el de hacer una aplicación más sencilla, simplemente focalizada en la planificación de la producción. Que en realidad es justamente lo que se busca: disminuir las prestaciones de forma de tener en cuenta solamente la necesidad de planificación del cliente, simplificando así la usabilidad y disminuyendo la complejidad de implementación y los costos.

Modelo

Modelo

Infraestructura

Tal como se comentó en el anteproyecto la infraestructura será WEB, con lo cual para su implementación se utilizará el framework Django sobre el sistema operativo Ubuntu 14.04 LTS. La base de datos a utilizar seguramente sea MySQL o PostgreSQL.

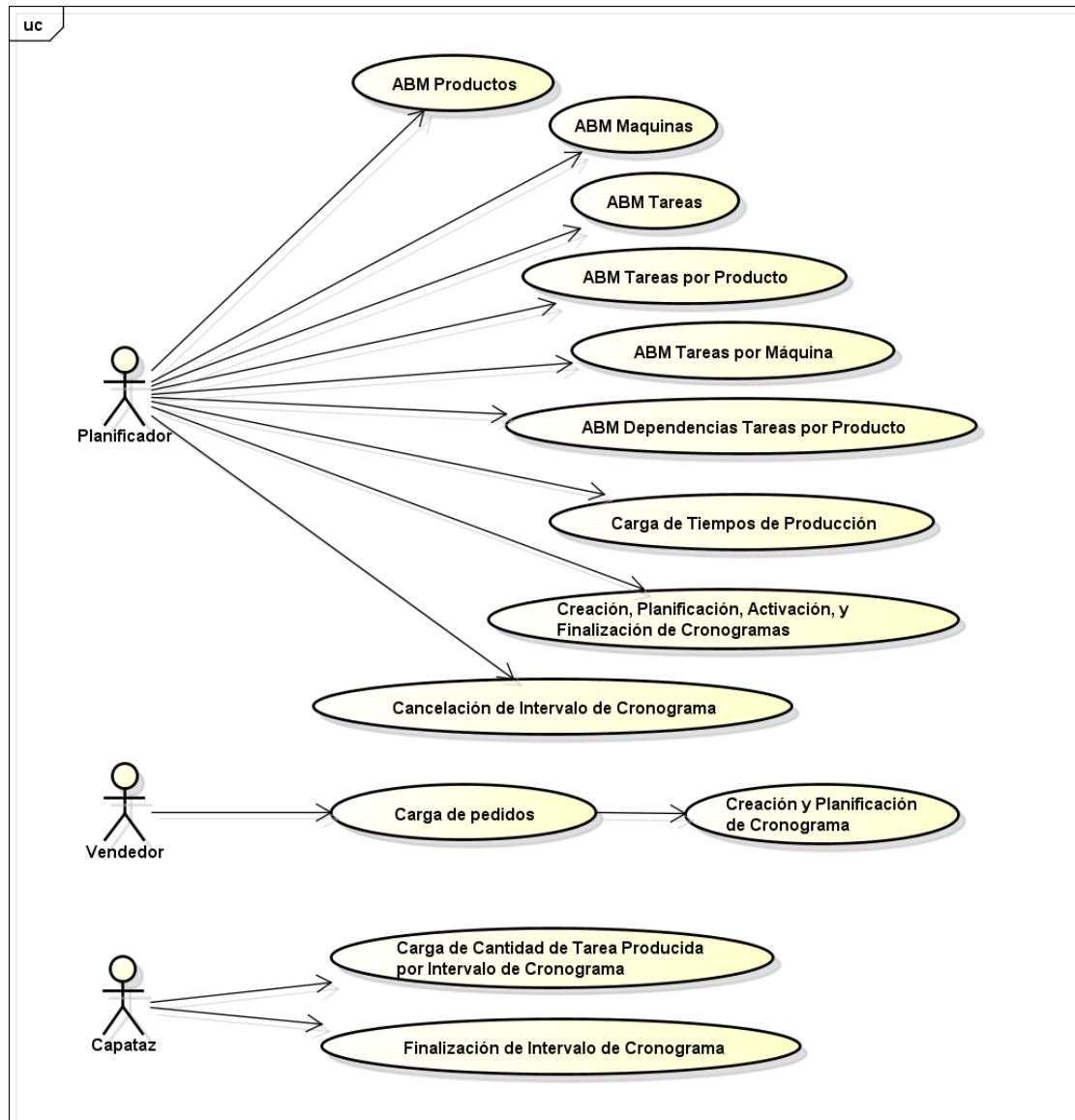
Como se puede observar la infraestructura a utilizar nos va a garantizar independencia, tanto a nivel de aplicación cliente (navegador web), como a nivel de licencias.

Por otro lado también se reducen costos al utilizar todas herramientas libres y de uso gratuito.

Casos de Uso

Diagrama de Casos de Uso

A continuación se muestra el correspondiente diagrama de casos de uso:



powered by Astah

Especificación de Casos de Uso

ABM Productos

Actor	Planificador
Descripción	Alta, baja y modificación de productos. Simplemente se crea un producto y se le asigna una descripción.
Flujo básico	<ol style="list-style-type: none"> 1) Se muestra listado de productos. 2) Agregar nuevo producto mostrará un formulario para cargar la información del nuevo producto. 3) Modificar producto mostrará un formulario para modificar el producto seleccionado. 4) Borrar producto eliminará un producto existente.

ABM Máquinas

Actor	Planificador
Descripción	Alta, baja y modificación de máquinas. Simplemente se crea una máquina y se le asigna una descripción.
Flujo básico	<ol style="list-style-type: none"> 1) Se muestra listado de máquinas. 2) Agregar nueva máquina mostrará un formulario para cargar la información de la nueva máquina. 3) Modificar máquina mostrará un formulario para modificar la máquina seleccionada. 4) Borrar máquina eliminará la máquina existente.

ABM Tareas

Actor	Planificador
Descripción	Alta, baja y modificación de tareas. Simplemente se crea una tarea y se le asigna una descripción y un tiempo de tarea por defecto.
Flujo básico	<ol style="list-style-type: none"> 1) Se muestra listado de tareas. 2) Agregar nueva tarea mostrará un formulario para cargar la información de la nueva tarea. 3) Modificar tarea mostrará un formulario para modificar la tarea seleccionada. 4) Borrar tarea eliminará la tarea existente.

ABM Tareas por Producto

Actor	Planificador
Descripción	Asignación de tareas que se realizan para crear un determinado producto.
Flujo básico	<ol style="list-style-type: none"> 1) Se selecciona producto. 2) Se muestra formulario donde se permite agregar y quitar tareas para el producto seleccionado

ABM Tareas por Máquina

Actor	Planificador
Descripción	Asignación de tareas que puede realizar una determinada máquina.
Flujo básico	<ol style="list-style-type: none"> 1) Se selecciona máquina. 2) Se muestra formulario donde se permite agregar y quitar tareas para la máquina seleccionada.

ABM Dependencias Tareas por Producto

Actor	Planificador
Descripción	<p>Dado un Producto se indica dependencias entre tareas.</p> <p>Sirve para generar un grafo de dependencias entre tareas y obviamente se validará que el grafo sea aciclico.</p>
Flujo básico	<ol style="list-style-type: none"> 1) Se selecciona producto. 2) Se muestra formulario donde se permite agregar y quitar dependencias. 3) Por cada dependencia agregada se podrá especificar 2 tareas, una será la anterior y otra la posterior. Está claro que la tarea posterior depende de la tarea anterior.

Carga de Tiempos de Producción

Actor	Planificador
Descripción	<p>Permitirá al usuario cargar el tiempo que se demora en realizar una Tarea para un Producto dado en una determinada Maquina.</p>
Flujo básico	<ol style="list-style-type: none"> 1) Se muestra listado con combinaciones posibles de las ternas (Tarea, Producto, Máquina) en función de la configuración cargada. 2) Se permitirá editar en dicho listado el tiempo de duración por cada terna posible y guardar los cambios realizados.

Creación, Planificación, Activación, y Finalización de Cronogramas

Actor	Planificador
Descripción	<p>Creación de cronogramas, planificación de los mismos, activación y finalización.</p>
Flujo básico	<ol style="list-style-type: none"> 1) Se muestra listado de cronogramas. 2) Agregar nuevo cronograma mostrará un formulario para cargar la información del cronograma. Información a cargar: <ol style="list-style-type: none"> a. Descripción. b. Pedidos a planificar. c. Máquinas a utilizar. d. Parámetros que indican cómo realizar la planificación. 3) Modificar cronograma mostrará un formulario para modificar el cronograma. Acciones posibles: <ol style="list-style-type: none"> a. Planificar: Planifica el cronograma en función de la configuración dada. b. Activar: Activa el cronograma y lo incluye en el calendario productivo. c. Finalizar: Marca todos los intervalos activos del cronograma como finalizados.

Cancelación de Intervalo de Cronograma

Actor	Planificador
Descripción	Cancela un intervalo activo de un cronograma y todos los intervalos dependientes.
Flujo básico	<ol style="list-style-type: none"> 1) Se selecciona un intervalo de un cronograma dado. 2) Se cancela el intervalo.

Carga de pedidos

Actor	Vendedor
Descripción	Carga de pedidos.
Flujo básico	<ol style="list-style-type: none"> 1) Se muestra listado de Pedidos. 2) Agregar un pedido muestra formulario para cargar la información de un pedido: <ol style="list-style-type: none"> a. Descripción b. Items: <ol style="list-style-type: none"> i. Producto ii. Cantidad 3) Modificar un pedido permite, modificar la información de un pedido. 4) Eliminar un pedido permite eliminar un pedido.

Creación y Planificación de Cronograma

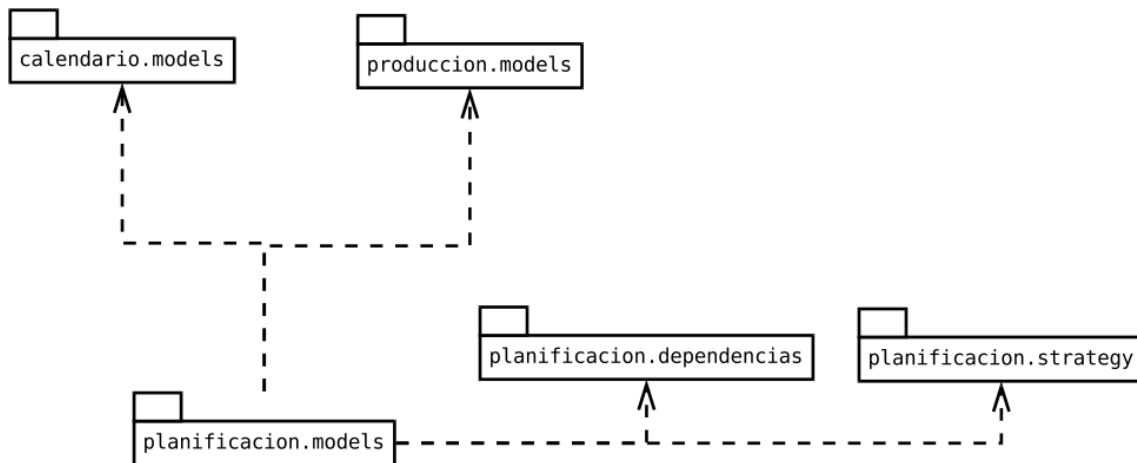
Actor	NA
Descripción	Una vez creado un pedido se crea un cronograma en forma automática con parámetros por defecto, se planifica y se activa.
Flujo básico	<ol style="list-style-type: none"> 1) Se crea cronograma y se incluye el pedido recién creado y todas las máquinas posibles asociadas a las tareas a realizar para fabricar los productos del pedido. 2) Se planifica el cronograma. 3) Se activa el cronograma.

Carga de Cantidad de Tarea Producida por Intervalo de Cronograma

Actor	Capataz
Descripción	Dado un intervalo activo que indica una cantidad de tarea a realizar, para un producto específico, en una determinada máquina, dentro de un intervalo de tiempo, se podrá cargar la cantidad de tarea real producida.
Flujo básico	<ol style="list-style-type: none"> 1) Se selecciona el intervalo. 2) Se ingresa la cantidad de tarea real producida. 3) Se guardan los cambios.

Finalización de Intervalo de Cronograma

Actor	Capataz
Descripción	Dado un intervalo activo que indica una cantidad de tarea a realizar, para un producto específico, en una determinada máquina, dentro de un intervalo de tiempo, se podrá marcar como finalizado dicho intervalo.
Flujo básico	<ol style="list-style-type: none"> 1) Se selecciona el intervalo activo. 2) Se marca como finalizado. 3) Si no se ingresó la cantidad de tarea real, se tomá por defecto que la misma es igual a la cantidad de tarea a producir.

Paquetes

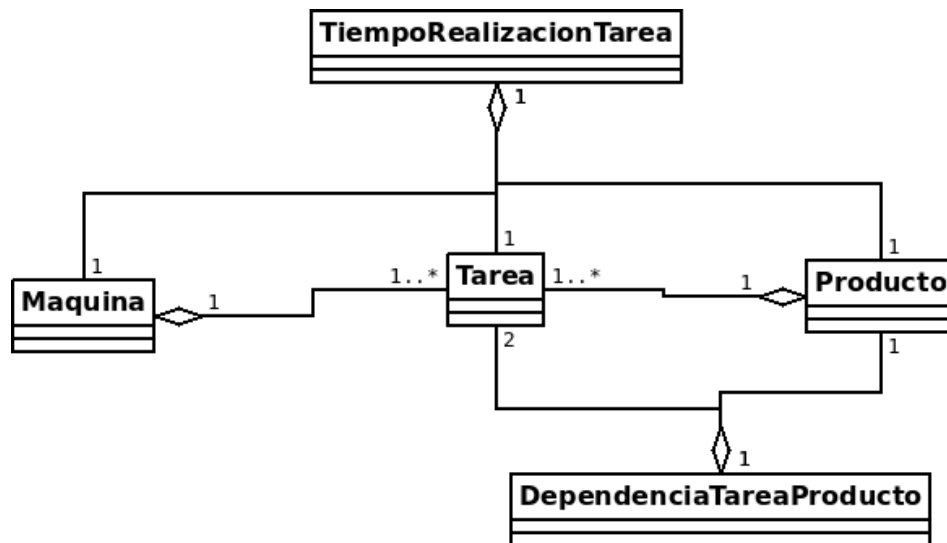
La idea principal es la de dividir la funcionalidad del modelo en 3 paquetes principales:

- producción: Es el modelo que describe los productos a fabricar, las tareas necesarias para llevar a cabo su fabricación, los recursos que realizan las tareas, los tiempos medios de cada tarea según producto/recurso, y las dependencias entre las tareas.
- planificación: Es el modelo que describe los pedidos a fabricar, que permite configurar cómo planificar cada pedido y que realiza la planificación. Dada la complejidad de este paquete la idea es dividirlo en tres subpaquetes:
 - models: Los modelos básicos de la planificación.
 - dependencias: Análisis y validación de dependencias en una planificación dada.
 - strategy: Estrategias posibles de planificación.
- calendario: Es el modelo que describe el calendario de producción. Permite configurar cuales son los intervalos laborables, y cuales son las excepciones laborables/no laborables.

Clases

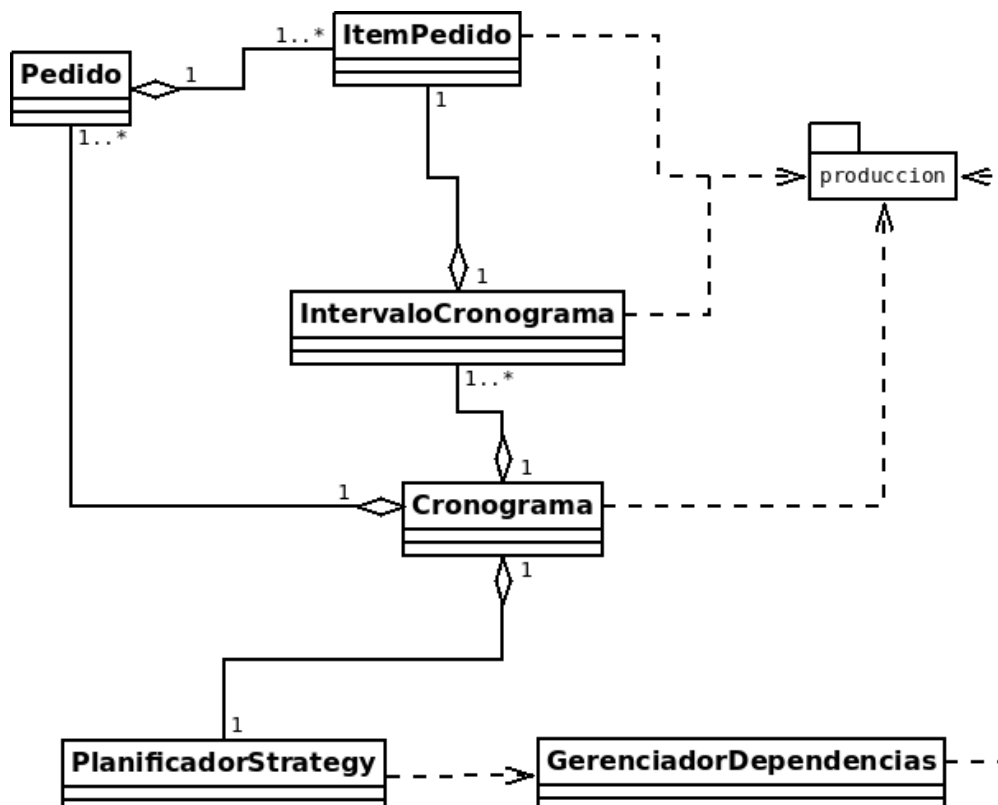
Paquete de producción

A continuación se muestran las principales clases del paquete de producción:



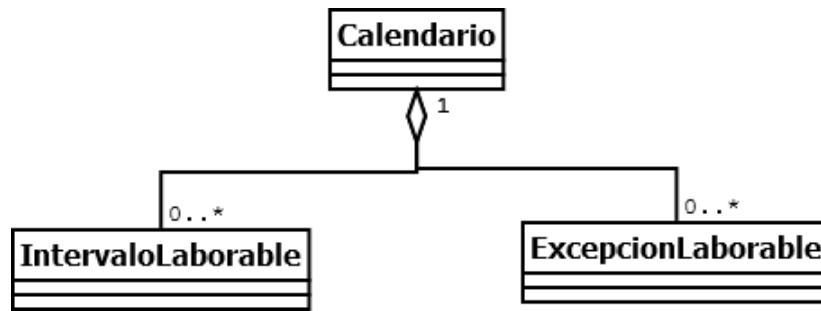
Paquete de planificación

A continuación se muestran las principales clases del paquete de planificación:



Paquete de calendario

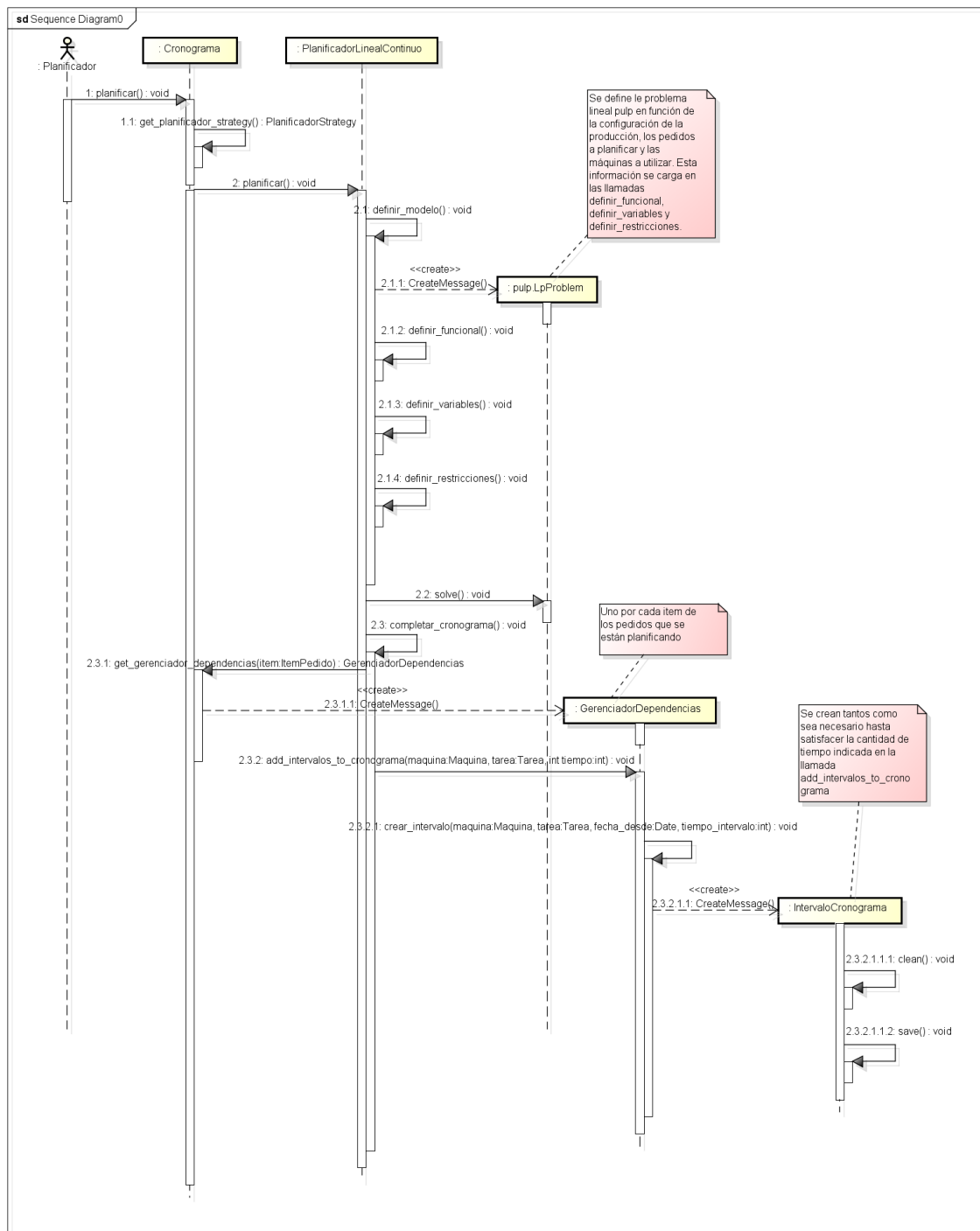
A continuación se muestran las principales clases del paquete de calendario:



Diagramas de Secuencia

Planificación de Cronograma

A continuación sigue el diagrama de secuencia que modela la planificación de un cronograma utilizando como estrategia un modelo de programación lineal:



Casos de Prueba

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))

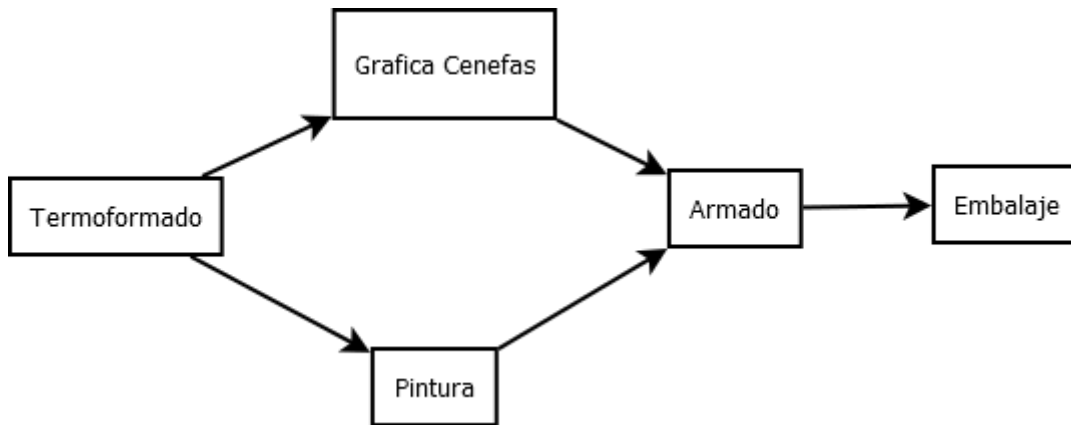
def verificar_dependencias(self, cronograma):
    """
    Se verifica que se respeten las dependencias entre las tareas.
    Básicamente la cantidad de tarea dependiente no puede superar
    en cantidad a la tarea de la cual depende.
    """
    for i in cronograma.get_intervalos():
        gerenciador_dependencias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

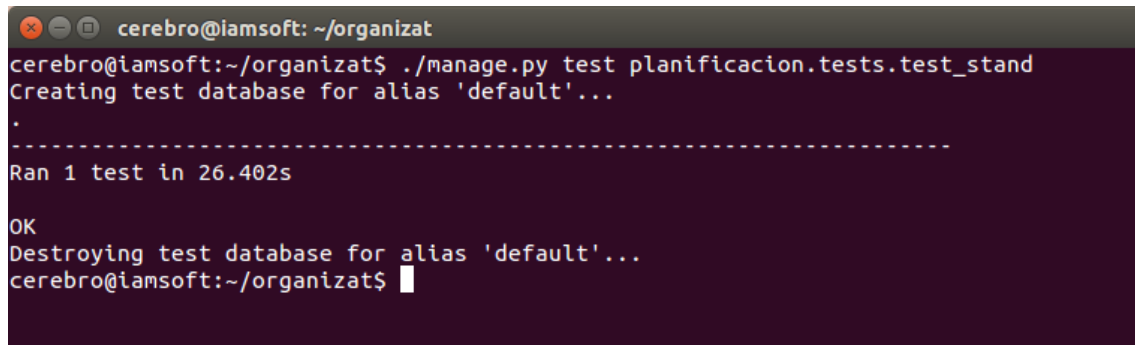
"""
Se verifica que se respeten las dependencias entre las tareas.
Básicamente la cantidad de tarea dependiente no puede superar
en cantidad a la tarea de la cual depende.
"""

self.verificar_dependencias(cronograma)

```

Ejecución del caso

A continuación se verifica una ejecución exitosa de la prueba:



```

cerebro@iamsoft: ~/organizat
cerebro@iamsoft:~/organizat$ ./manage.py test planificacion.tests.test_stand
Creating test database for alias 'default'...
.
-----
Ran 1 test in 26.402s

OK
Destroying test database for alias 'default'...
cerebro@iamsoft:~/organizat$

```

Resultado de la ejecución

A continuación se listan los intervalos de producción creados a partir de la ejecución de la prueba:

```

[#1] [Termoformadora 1] [Termoformado] [P#1-I#1-Stand] [cant: 8] [2014-12-04
08:00]-[2014-12-04 12:00]

[#2] [Termoformadora 1] [Termoformado] [P#1-I#1-Stand] [cant: 8] [2014-12-04
13:00]-[2014-12-04 17:00]

[#3] [Termoformadora 1] [Termoformado] [P#1-I#1-Stand] [cant: 8] [2014-12-05
08:00]-[2014-12-05 12:00]

[#4] [Termoformadora 1] [Termoformado] [P#1-I#1-Stand] [cant: 1] [2014-12-05
13:00]-[2014-12-05 13:30]

[#5] [Termoformadora 2] [Termoformado] [P#1-I#1-Stand] [cant: 8] [2014-12-04
08:00]-[2014-12-04 12:00]

[#6] [Termoformadora 2] [Termoformado] [P#1-I#1-Stand] [cant: 8] [2014-12-04
13:00]-[2014-12-04 17:00]

[#7] [Termoformadora 2] [Termoformado] [P#1-I#1-Stand] [cant: 8] [2014-12-05
08:00]-[2014-12-05 12:00]

[#8] [Termoformadora 2] [Termoformado] [P#1-I#1-Stand] [cant: 1] [2014-12-05
13:00]-[2014-12-05 13:30]

[#9] [Plotter] [Grafica Cenefas] [P#1-I#1-Stand] [cant:
16.07916666666666571927635232] [2014-12-04 08:47]-[2014-12-04 12:00]

[#10] [Plotter] [Grafica Cenefas] [P#1-I#1-Stand] [cant:
16.07916666666666571927635232] [2014-12-04 13:47]-[2014-12-04 17:00]

[#11] [Plotter] [Grafica Cenefas] [P#1-I#1-Stand] [cant:
16.07916666666666571927635232] [2014-12-05 08:47]-[2014-12-05 12:00]

[#12] [Plotter] [Grafica Cenefas] [P#1-I#1-Stand] [cant:
1.7625000000000002842170943040] [2014-12-05 13:05]-[2014-12-05 13:26]

[#13] [Cabina Pintura] [Pintura] [P#1-I#1-Stand] [cant: 10] [2014-12-04
08:00]-[2014-12-04 12:00]

[#14] [Cabina Pintura] [Pintura] [P#1-I#1-Stand] [cant: 10] [2014-12-04
13:00]-[2014-12-04 17:00]

[#15] [Cabina Pintura] [Pintura] [P#1-I#1-Stand] [cant: 10] [2014-12-05
08:00]-[2014-12-05 12:00]

```

[#16] [Cabina Pintura] [Pintura] [P#1-I#1-Stand] [cant: 10] [2014-12-05 13:00]-[2014-12-05 17:00]

[#17] [Cabina Pintura] [Pintura] [P#1-I#1-Stand] [cant: 10] [2014-12-08 08:00]-[2014-12-08 12:00]

[#18] [Mano de Obra] [Armado] [P#1-I#1-Stand] [cant: 10.04907407407407365301171214] [2014-12-04 08:59]-[2014-12-04 12:00]

[#19] [Mano de Obra] [Armado] [P#1-I#1-Stand] [cant: 10.05092592592592539959797351] [2014-12-04 13:59]-[2014-12-04 17:00]

[#20] [Mano de Obra] [Armado] [P#1-I#1-Stand] [cant: 10.04907407407407365301171214] [2014-12-05 08:59]-[2014-12-05 12:00]

[#21] [Mano de Obra] [Armado] [P#1-I#1-Stand] [cant: 10.05092592592592539959797351] [2014-12-05 13:59]-[2014-12-05 17:00]

[#22] [Mano de Obra] [Armado] [P#1-I#1-Stand] [cant: 9.800000000000003473764485938] [2014-12-08 08:57]-[2014-12-08 11:54]

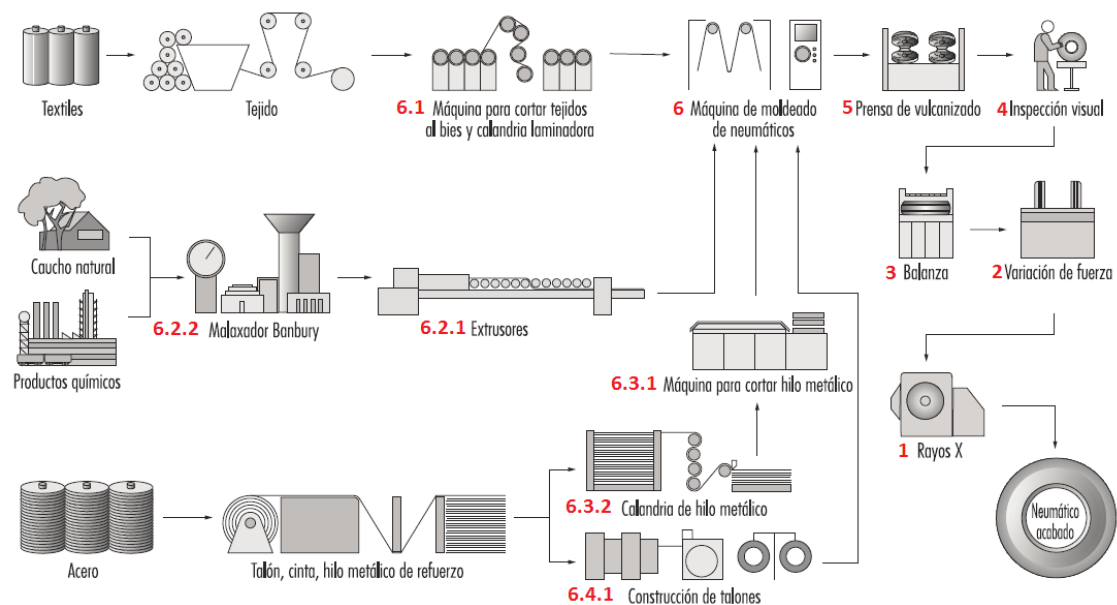
[#23] [Mano de Obra] [Embalaje] [P#1-I#1-Stand] [cant: 26.666666666666666666666666666667] [2014-12-08 13:00]-[2014-12-08 17:00]

[#24] [Mano de Obra] [Embalaje] [P#1-I#1-Stand] [cant: 23.333333333333333333333333333333] [2014-12-09 08:00]-[2014-12-09 11:30]

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 - Soporte 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_extrusion_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_extrusion_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_extrusion_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)
```

Ejecución del caso

A continuación se verifica una ejecución exitosa de la prueba:

```
cerebro@iamsoft: ~/organizat
cerebro@iamsoft:~/organizat$ ./manage.py test planificacion.tests.test_neumatico
Creating test database for alias 'default'...
.
-----
Ran 1 test in 16.699s

OK
Destroying test database for alias 'default'...
cerebro@iamsoft:~/organizat$
```

Resultado de la ejecución

A continuación se listan los intervalos de producción creados a partir de la ejecución de la prueba:

[#1] [6.1 - Cortadora Laminadora] [6.1.2 - Laminado Tejido] [P#1-I#1-Neumático] [cant: 2.0E+2] [2014-12-04 08:00]-[2014-12-04 09:20]

[#2] [6.2.2 - Malaxador Banbury] [6.2.2 - Obtención Caucho] [P#1-I#1-Neumático] [cant: 2.0E+2] [2014-12-04 08:00]-[2014-12-04 09:20]

[#3] [6.3.2 - Calandria Laminadora de Hilo Metálico] [6.3.2 - Laminado Hilo Metálico] [P#1-I#1-Neumático] [cant: 2.0E+2] [2014-12-04 08:00]-[2014-12-04 09:20]

[#4] [6.4.1 - Constructora de Talones] [6.4.1 - Construcción de Talones] [P#1-I#1-Neumático] [cant: 120] [2014-12-04 08:00]-[2014-12-04 12:00]

[#5] [6.4.1 - Constructora de Talones] [6.4.1 - Construcción de Talones] [P#1-I#1-Neumático] [cant: 80] [2014-12-04 13:00]-[2014-12-04 15:40]

[#6] [6.1 - Cortadora Laminadora] [6.1.1 - Cortado Tejido] [P#1-I#1-Neumático]
[cant: 2.0E+2] [2014-12-04 09:20]-[2014-12-04 10:00]

[#7] [6.2.1 - Extructura de Caucho] [6.2.1 - Extrucción Caucho] [P#1-I#1-Neumático] [cant: 2.0E+2] [2014-12-04 08:00]-[2014-12-04 10:00]

[#8] [6.3.1 - Cortadora de Hilo Metálico] [6.3.1 - Cortado Hilo Metálico]
[P#1-I#1-Neumático] [cant: 2.0E+2] [2014-12-04 08:59]-[2014-12-04 09:19]

[#9] [6 - Moldeadora de Neumáticos 1] [6 - Moldeado] [P#1-I#1-Neumático]
[cant: 79.99166666666666003493446624] [2014-12-04 09:20]-[2014-12-04 12:00]

[#10] [6 - Moldeadora de Neumáticos 1] [6 - Moldeado] [P#1-I#1-Neumático]
[cant: 20.00833333333333996506553376] [2014-12-04 13:00]-[2014-12-04 13:40]

[#11] [6 - Moldeadora de Neumáticos 2] [6 - Moldeado] [P#1-I#1-Neumático]
[cant: 40.59166666666666856144729536] [2014-12-04 10:38]-[2014-12-04 12:00]

[#12] [6 - Moldeadora de Neumáticos 2] [6 - Moldeado] [P#1-I#1-Neumático]
[cant: 59.40833333333333143855270464] [2014-12-04 13:39]-[2014-12-04 15:38]

[#13] [5 - Prensa de Vulcanizado] [5 - Prensado] [P#1-I#1-Neumático] [cant:
93.711111111111110858473693952] [2014-12-04 09:39]-[2014-12-04 12:00]

[#14] [5 - Prensa de Vulcanizado] [5 - Prensado] [P#1-I#1-Neumático] [cant:
106.2888888888888914152630605] [2014-12-04 13:00]-[2014-12-04 15:39]

[#15] [4 - Soporte Para Inspección] [4 - Inspección visual] [P#1-I#1-Neumático]
[cant: 2.0E+2] [2014-12-04 13:57]-[2014-12-04 15:37]

[#16] [3 - Balanza] [3 - Pesaje] [P#1-I#1-Neumático] [cant: 2E+2] [2014-12-04
15:07]-[2014-12-04 15:37]

[#17] [2 - Variador de Fuerza] [2 - Pruebas de fuerza] [P#1-I#1-Neumático]
[cant: 56.27499999999999857891452848] [2014-12-04 15:07]-[2014-12-04 17:00]

[#18] [2 - Variador de Fuerza] [2 - Pruebas de fuerza] [P#1-I#1-Neumático]
[cant: 120] [2014-12-05 08:00]-[2014-12-05 12:00]

[#19] [2 - Variador de Fuerza] [2 - Pruebas de fuerza] [P#1-I#1-Neumático]
[cant: 23.72499999999999431565811392] [2014-12-05 13:00]-[2014-12-05 13:47]

[#20] [1 - Rayos X] [1 - Verificación final] [P#1-I#1-Neumático] [cant:
2.0E+2] [2014-12-05 13:05]-[2014-12-05 13:45]

Futuras Líneas de Trabajo

Futuras Líneas de Trabajo

Hasta el día de hoy se llegó a desarrollar producto que cumple el objetivo planteado, sin embargo para poder llegar a una etapa productiva es fundamental atacar las siguientes ramas de trabajo:

1. Usabilidad de usuario.
2. Implementación de calendario de máquina.
3. Incorporación de materia prima al proceso productivo.
4. Incorporación de costos de producción.

Usabilidad de usuario

Esta rama es fundamental y la que merecería trabajo inmediato, ya que mejorando este aspecto de la aplicación, ya podríamos pasar a una etapa productiva e incorporar el producto a diferentes clientes. Esto permitirá detectar e incorporar mejoras rápidamente dependiendo de las necesidades detectadas en cada cliente.

Implementación de calendario de máquina

Incorporando un calendario por máquina, en lugar de utilizar un único calendario de fábrica, permitirá, por ejemplo, planificar mantenimientos preventivos, incorporar la rotura de una máquina y realizar la posterior re planificación de la producción.

Esto, que es moneda corriente en una fábrica, es primordial incorporarlo de forma de poder modelar correctamente los cronogramas de producción.

Incorporación de materia prima al proceso productivo

Como funcionalidad adicional a la planificación de la producción, sería muy útil incorporar un flujo de materia prima día a día.

Aprovechando la información de la planificación de la producción, y conociendo la materia prima necesaria para cada tarea, sería bastante útil poder ofrecer al cliente, la información de la cantidad de materia prima necesaria por cada intervalo de producción. Luego se podría indicar la cantidad de materia prima por períodos (día, semana, quincena, etc.), de forma que sirva para el proceso de compras y planificación de almacenamiento.

Incorporación de costos de producción

Ya teniendo la planificación de la producción y materia prima necesaria por cada intervalo productivo, si se incorpora en la configuración el costo por unidad de tiempo de cada recurso y el costo de materia prima por unidad utilizada, entonces podríamos planificar el costo de producción.

A partir de la información de costos, se podría ofrecer al cliente un flujo de caja de costos de forma que sirva para prever la situación financiera.

Bibliografía y Referencias

Bibliografía

- Documentación Pulp: <https://pulp-user-guide.readthedocs.org/en/pulp-2.4/>
- Documentación Django: <https://docs.djangoproject.com/en/1.6/>
- Documentación Python: <https://www.python.org/doc/>
- ENCICLOPEDIA DE SALUD Y SEGURIDAD EN EL TRABAJO; Capitulo 80: INDUSTRIA DEL CAUCHO; Proceso de fabricación de neumáticos; Alan Echt.

Referencias

- Presea (NeuralSoft): <https://www.neuralsoft.com/productos/presea>
- PlannerOne (Ortems): <http://www.plannerone.com/?lang=es>
- Super VISION (Lemdi): <http://www.lemdi.com/es/p5-software.html>
- Preactor (SIMLog): <http://www.simlog.com.ar/>
- Página Web del Autor: <http://juan.sebastian.goldberg.iamsoft.com.ar/>