Trabajo Práctico Final

Bruno, Tomás ; *Padrón Nro. 88.449*
tbruno88@gmail.com
Ferreiro, Demian ; *Padrón Nro. 88443*
epidemian@gmail.com
Leguizamo, Matías ; *Padrón Nro.*
matias.leguizamo@gmail.com
Mouso, Nicolás ; *Padrón Nro. 88.528*
nicolasgnr@gmail.com

Ayudante: Emiliano Ritiro
1er. Cuatrimestre de 2010
75.10 Tecnicas de Diseño
Facultad de Ingeniería, Universidad de Buenos Aires

Trabajo Práctico Final

Técnicas de diseño

April 21, 2010

1 Objetivo

Desarrollar una aplicacion de escritorio con interfaz gráfica que permita integrar y profundizar los conocimientos de diseño y programación orientada a objetos adquiridos en la materia, aplicandolos en un trabajo práctico grupal.

2 Enunciado

Un grupo de ingenieros industriales analizó la necesidad de ampliar la cantidad de ingresantes a la carrera y, dado que proyectaron a largo plazo, decidieron invertir en un video juego que captara la atencion de los mas jovenes en la disciplina industrial. Para ello, se opto por desarrollar un simulador de fabricas en el cual el jugador puede crear su propia planta industrial a gusto y piacere, acorde a su presupuesto y las dimensiones del terreno donde situará la fabrica.

2.1 Dinamica del juego

El jugador arrancara el juego en determinada fecha con un dinero inicial y a partir de entonces cada determinada cantidad de segundos avanzara un dia en el calendario virtual. Al terminar cada dia, se incorporara al dinero del jugador la diferencia entre el dinero ganado por los productos producidos y los costos insumidos por producirlos (materias primas) mas los costos de mantenimiento de la fabrica (alquiler del terreno, etc.). La lista de costos de las materias primas y la lista de precios de los productos varía todas las semanas acorde a un xml de entrada. El jugador tendra la capacidad de frenar en cualquier momento el paso del tiempo para poder hacer cambios en la estructura de su fabrica (comprar nuevas maquinas, moverlas de lugar, realizar reparaciones, etc.) sin tener que pagar los costos de tener la linea de produccion parada.

2.2 Inicio del juego

El juego se inicia con una pantalla donde el jugador podra elegir que terreno alquilar o comprar para instalar su fabrica. Al arrancar, el sistema se encontrara en modo pausa permitiendole al usuario construir una linea de produccion inicial antes de comenzar a pagar los gastos. Para ello el jugador debera comprar maquinas utilizando su presupuesto inicial. Estas se conectarán mediante cintas transportadoras siguiendo ciertos pasos acordes al producto que se quiere desarrollar.

2.3 Maquinas y cintas transportadoras

Cada maquina posee una entrada y una salida. Para poder utilizarla, el jugador debera conectar la entrada a la salida de otra maquina mediante una cinta transportadora. La primer maquina de la linea estará exenta de esta regla, ya que utilizara unicamente como entrada la materia prima elegida. La última maquina de la linea estara conectada a la salida de la fabrica mediante una cinta transportadora. La cinta tendra un costo por metro y las maquinas un costo por unidad.

Cada maquina ocupara un espacio fisico que no puede ser utilizado por ninguna otra, ni por una cinta transportadora. Lo mismo vale para las cintas. Cada cinta tiene un sentido y ese sentido debe ser validado: En un extremo tendra la salida de una maquina y en el otro la entrada y la cinta circulará en esa dirección.

Toda maquina puede ser reubicada dentro de la fábrica. No es necesario que la cinta transportadora lo sea, pero tanto maquinas como cintas deben poder ser eliminadas. Estas operaciones seran gratuitas, es decir, el jugador no pagara nada por reubicar una maquina o borrar una cinta.

En caso de que el usuario construya una linea de produccion ciclica, el sistema deberá detectarlo y al iniciar la producción las maquinas de la linea tendrán que romperse.

2.4 Productos

Cada producto para ser desarrollado requiere de un proceso en particular determinado por las maquinas que componen su linea de produccion. Una fabrica puede elaborar multiples productos, limitada unicamente por el espacio físico de la misma.

`Ej de fabricación: Para producir ceramicas es necesario: Ingresar materia prima a una prensadora, llevar la pieza prensada a un horno. La salida del horno deberá estar conectada con una linea de esmaltado. Opcionalmente pasará por una máquina detectora de piezas defectuosas. Finalmente terminará en una máquina empaquetadora que apilará y guardará en cajas las piezas fabricadas.`

Para simplificar el problema, todas las maquinas producen a la misma velocidad y la cantidad de piezas del producto realizadas por dia es constante. Al finalizar el dia se computan las piezas buenas producidas, las piezas defectuosas y en base a eso y los costos del mercado se calcula la ganancia diaria.
Si el jugador crea una linea de producción que no produce ningun producto conocido (o que aun no se ha descubierto) esta producirá desechos. Producir desechos genera el costo de la materia prima insumida pero no da ningun redito economico. No confundir desechos con piezas defectuosas: Los desechos no alteran la calidad de la fabrica.

2.5 Averías

Toda maquina es factible de averiarse. Algunas con mayor probabilidad que otras. Azarosamente alguna de las maquinas se rompera o comenzara a producir piezas con menor calidad (mayor probabilidad de productos defectuosos). En ese caso el jugador puede poner pausa y arreglar la maquina averiada con un costo proporcional al precio de la misma. Las cintas no se averían.

Si una maquina se rompe, toda esa linea de produccion quedará frenada. Con lo cual el jugador dejara de percibir ingresos por esa linea.

2.6 Calidad

Todo producto puede pasar opcionalmente por un proceso de control de calidad mediante la instalacion de una maquina para tal fin. Azarosamente, un numero de las piezas producidas seran defectuosas, con lo cual si se omite este paso, llegara al mercado un mayor porcentaje de productos de mala calidad. El porcentaje de piezas defectuosas harán que los consumidores tengan una vision mas o menos positiva de la fabrica. Por lo tanto, el precio de venta estara atado adicionalmente a la imagen que se tenga de la fabrica. para simplificar se usara la siguiente regla:
(1 - [% de piezas defectuosas])^2 X [Precio del producto en el mercado] = [Precio por pieza producida].

`Ej: Una fabrica cuyo porcentaje de piezas defectuosas es del 20% cobrara por un producto que vale $10:`
`(1 - 0.2)`^{2} `x 10 = 6.4 por pieza producida.`

El costo de calidad es universal para todos los productos de la fabrica, es decir, el porcentaje de piezas defectuosas se calcula en forma global para todos los productos que produce la fabrica.

2.7 Terreno

Si bien al inicio del juego el jugador podra seleccionar un terreno para comprar o alquilar, su presupuesto inicial le impedira comprar un terreno caro. Es por eso que el jugador en cualquier momento puede vender su fabrica e iniciar una nueva en un predio mas grande.
Un jugador siempre utilizará un solo terreno a la vez. Si no desea comprarlo puede alquilarlo, en cuyo caso tendra que pagar determinado dia de cada mes una suma fija.
Al vender una fabrica se le reintegrará el 50% del valor de las maquinas que no esten rotas y en caso de ser propietario del terreno, el 80% del valor del mismo.

2.8 Laboratorio

El jugador tiene la opción de invertir parte de su redito diario en un laboratorio de desarrollo. Transcurrida cierta cantidad de dias nuevos productos o nuevas maneras de desarrollarlos estaran disponibles para el jugador. La frecuencia con el que el laboratorio hace sus descubrimientos es proporcional al

dinero invertido en investigación. Teniendo este un tope. Las maneras de producir productos deben poder ser configurables mediante un xml. El laboratorio "habilita" productos existentes en el xml, considerando aquellos productos con capital de inversion cero como disponibles desde el inicio de la partida.

2.9 Fin de juego

El jugador pierde cuando entra en banca rota (su dinero se acabó y la fabrica tiene un balance diario negativo) y gana al cumplir un objetivo monetario (posee una cierta cantidad de dinero ahorrada).

3 Consideraciones finales y alcance

Para nuestro cliente (Grupo de Ingenieros Industriales S.A.) lo mas importante es que el sistema sea flexible y configurable en los puntos mencionados para que ellos puedan analizarlo bien antes de salir al mercado.

Se plantea una segunda fase de desarrollo en la cual se hará incapie en la parte grafica y se extenderá la funcionalidad del juego. Por lo tanto, para el cliente no es importante que la aplicacion se vea bonita en esta primera fase, aunque si debe permitirle poder validar que todas las reglas de juego fueron correctamente implementadas.

Índice

1. Hipotésis

1. Las máquinas se pueden reparar por separado o se puede reparar toda una línea al mismo tiempo
2. Se pueden romper tanto las máquinas de control de calidad como las que son exclusivamente para la producción.
3. Tanto los precios de la materia prima como los de los productos varían todas las semanas. El resto de los precios se mantiene invariante en el tiempo
4. Las paredes no pueden ser borradas ni vendidas del terreno adquirido.
5. Los productos son construidos a partir de distintos tipos de materias primas y de cantidades de cada una de estas. Pueden suceder que existan dos líneas de producción que pese a tener una misma secuencia de producción, tengan como resultado productos distintos como consecuencia de tener una entrada de materia prima distinta.
6. Para poder producir, cada línea de producción debe estar conectada a una salida de la fabrica.
7. El precio de alquiler de la fabrica es proporcional al precio del terreno.
8. Las maquinas pueden no contener una entrada y una salida determinada en el archivo XML. En caso de que no tengan una establecida entonces se les genera una por defecto.
9. Tanto las salidas como las paredes de las fabricas se encuentran declaradas en el archivo XML correspodiente a los distintos terrenos.
10. Las máquinas de control de calidad pueden convertir a un producto en defectuoso.
11. Puede suceder que un mismo tipo de producto sea creado a partir de distintas secuencias de producción y diferentes materiales.

2. Vista Lógica

2.1. Producción

La forma en la cual el juego simula la producción es el núcleo del modelo.

La primer decisión de relevancia fue si en el juego cada producto iba a ser un objeto que fuera circulando por cada maquina o si simplemente al final del día se discretizaría la producción diaria. Decidimos optar por la primer alternativa porque consideramos que la segunda alternativa no representaba correctamente al negocio. Aunque se podrían modelizar las funcionalidades solicitadas en el presente, cada futuro cambio llevaría a nuevos parches con peligro de tener que desechar el núcleo del modelo por completo en caso de que no quedara otra alternativa. La segunda alternativa probablemente sería una buena opción en caso de que se tuvieran que manejar varias fabricas simultáneamente.

Una vez decidido acerca de la existencia fisica de cada producto, se abrieron distintos interrogantes. Un ejemplo fue la forma en la cual se identificaría el producto luego de ser procesado por las distintas máquinas.

La primera alternativa que se considero fue que cada línea fuera la encargada de identificar el tipo de producto que producía. Esto parecía lógico ya que la línea sabe las máquinas que tiene. Sin embargo, identificamos una limitación en esta solución. Esta limitación surgía de que cada vez que modificaramos las máquinas de la línea, los productos que estuvieran siendo procesados en ese momento en la línea podrían terminar resolviendo su tipo incorrectamente.

En consecuencia, decidimos que cada producto lleve un historial de las máquinas por las que pasaba. Al final de ser procesado por todas las máquinas, se analizaba cual era el producto final.

Con el objetivo de aumentar el interes al momento de jugarlo, decidimos tomar como hipotésis que el producto puede necesitar distintas materias primas y cantidades de cada una de ellas.

En el diagrama de clases se puede observar a grandes rasgos como se implementó la decisión de que cada producto guarda un historial: cada producto cuando es procesado por una maquina guarda la información acerca del tipo de maquina por el que fue procesado además de la materia de prima con la que fue creado . Para que esto sea realizado, existe un método process que esta redefinido en los distintos tipos de productionLineElements.

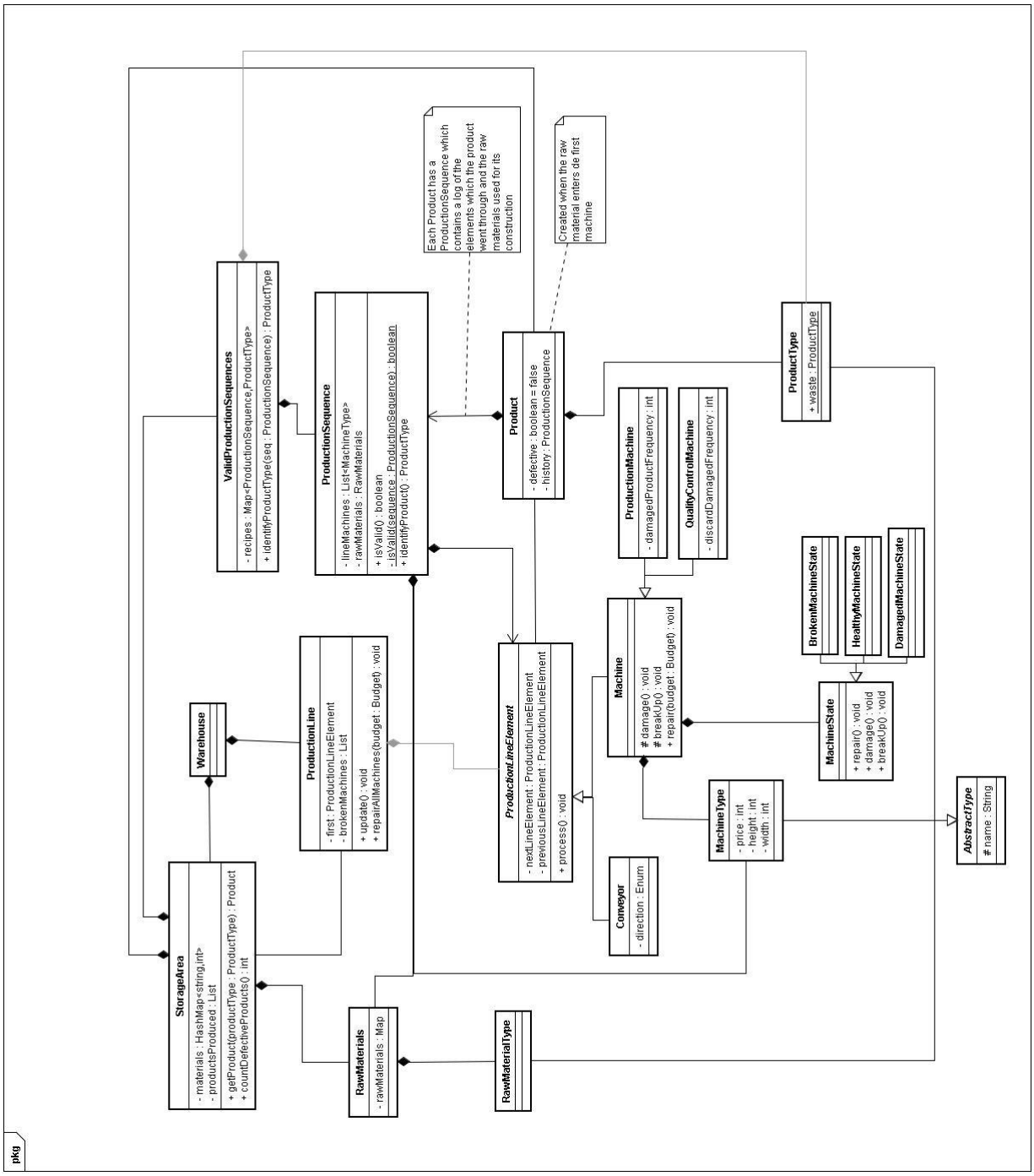

Figura 1: Diagrama de Clases de la lógica de producción

En el diagrama de secuencia, se puede observar como es la secuencia desde que sea crea un producto amorfo, que no sabe que tipo de producto es, hasta que finalmente se resuelve su tipo. En el interin, el producto es procesado por

todos los distintos elementos que forman parte de la línea de producción.

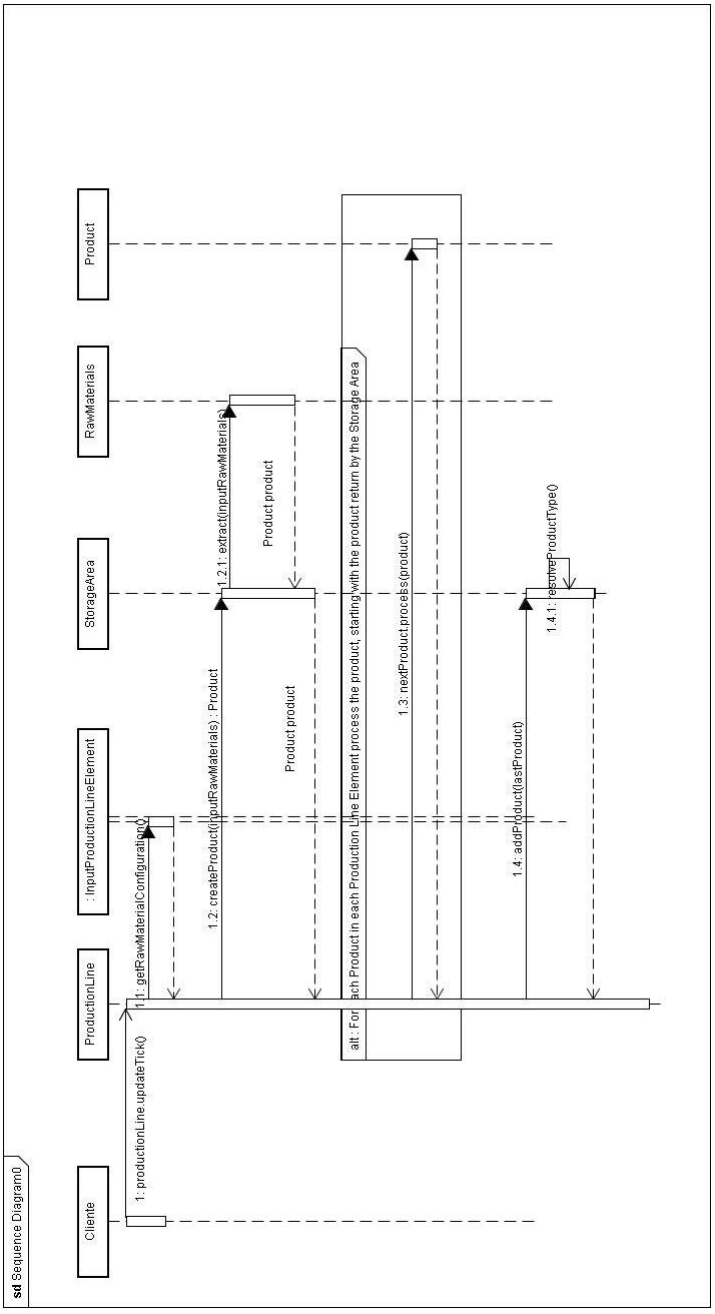

Figura 2: Diagrama de Secuencia de la lógica de producción

2.2. Discretización Temporal

Como se podía observar en el grafico , existe un metodo en la lńea de producción que dispara la circulación del producto a través de las máquinas. En cada “tick”, un producto pasa de una maquinaa la siguiente. De la misma forma, existen otras entidades que se suscriben a una entidad que se encarga de actualizar el paso del tiempo en todos los objetos que esten suscriptos a ella. Cada día esta formado por una cantidad especificada de ticks. A su vez, la cantidad de días por semana y días por mes tambíen se encuentra parametizada.

2.3. Laboratorio de Tecnologías

El laboratorio se ocupa de desarrollar las distintas secuencias de producción que resultan en productos validos. Cada desarrollo que realiza el laboratorio se considera una *tecnología* y cada una de las cuales habilita o una nueva forma de producir un producto o un producto nuevo.

Al momento de diseñar el laboratorio se busco que el laboratorio no sea algo secundario y prescindible. Así fue que se decidio que las tecnologías que este desarrolla tengan una estructura jerarquica en la cual cada tecnología puede tener dependencias. Es decir, que se necesite desarrollar otras tecnologías antes de esta.

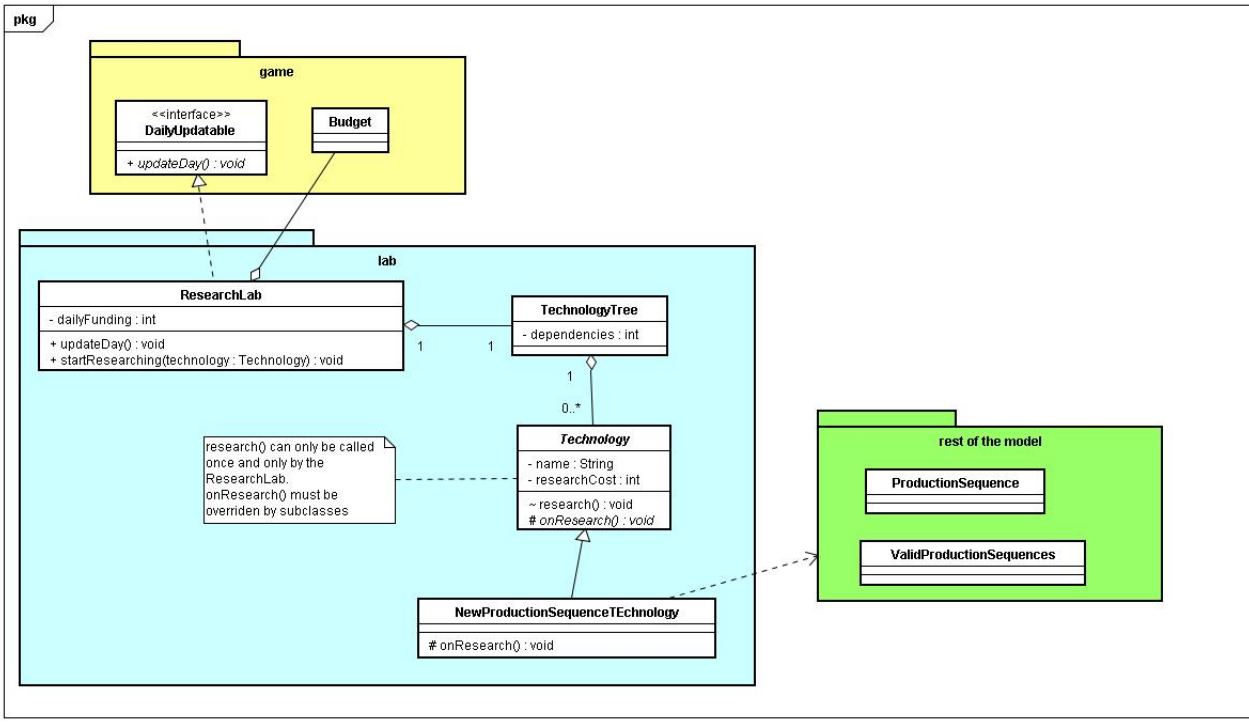

Figura 3: Diagrama de Clases de la lógica del desarrollo de Tecnologías

2.4. Arquitectura General

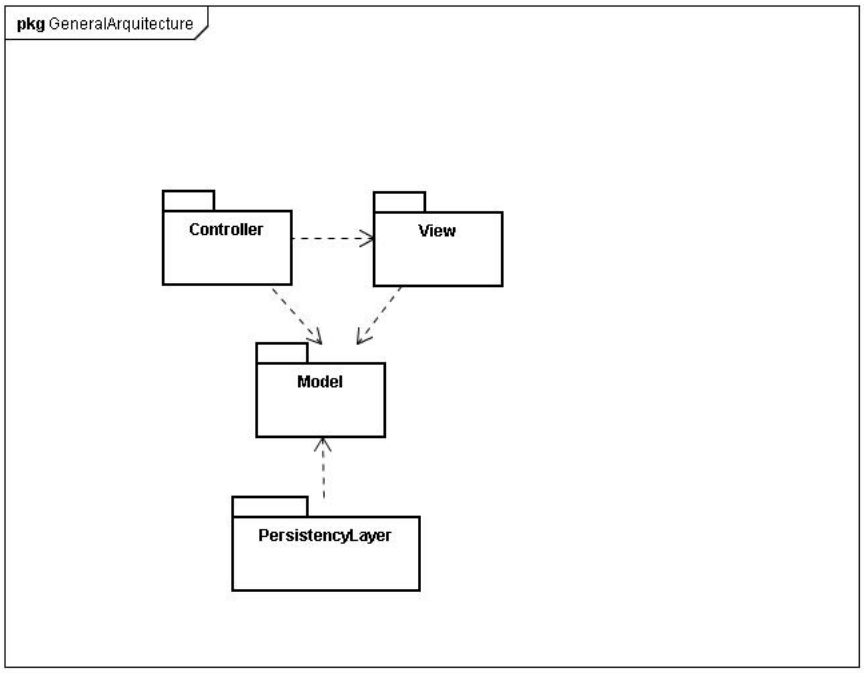

Figura 4: Diagrama de Paqutes general de la Arquitectura

La arquitectura sigue al patrón de arquitectura, *MVC*. El modelo no depende de nada más que si mismo, Los controladores utilizan al modelo y a la vista y la vista necesita del modelo. La capa de persistencia también esta desacoplada completamente del modelo.

2.5. Persistencia

Como ya se explico anteriormente, la persistencia quedo desacoplada del modelo. Esto se puede observar en el diagrama de clases, donde se ve como existe una interfaz que es necesario implementar para la recuperación de datos que son indispensables para el modelo.

Esta interfaz, es implementada por una clase concreta ,*XMLFactory* ,que se encuentra dentro del paquete de persistencia. Aca es donde se encuentra el hecho de que la recuperación es sobre XMLS y no por ejemplo, sobre una base de datos. Fácilmente se podría intercambiar el *input* del sistema por otro que venga en una base de datos.

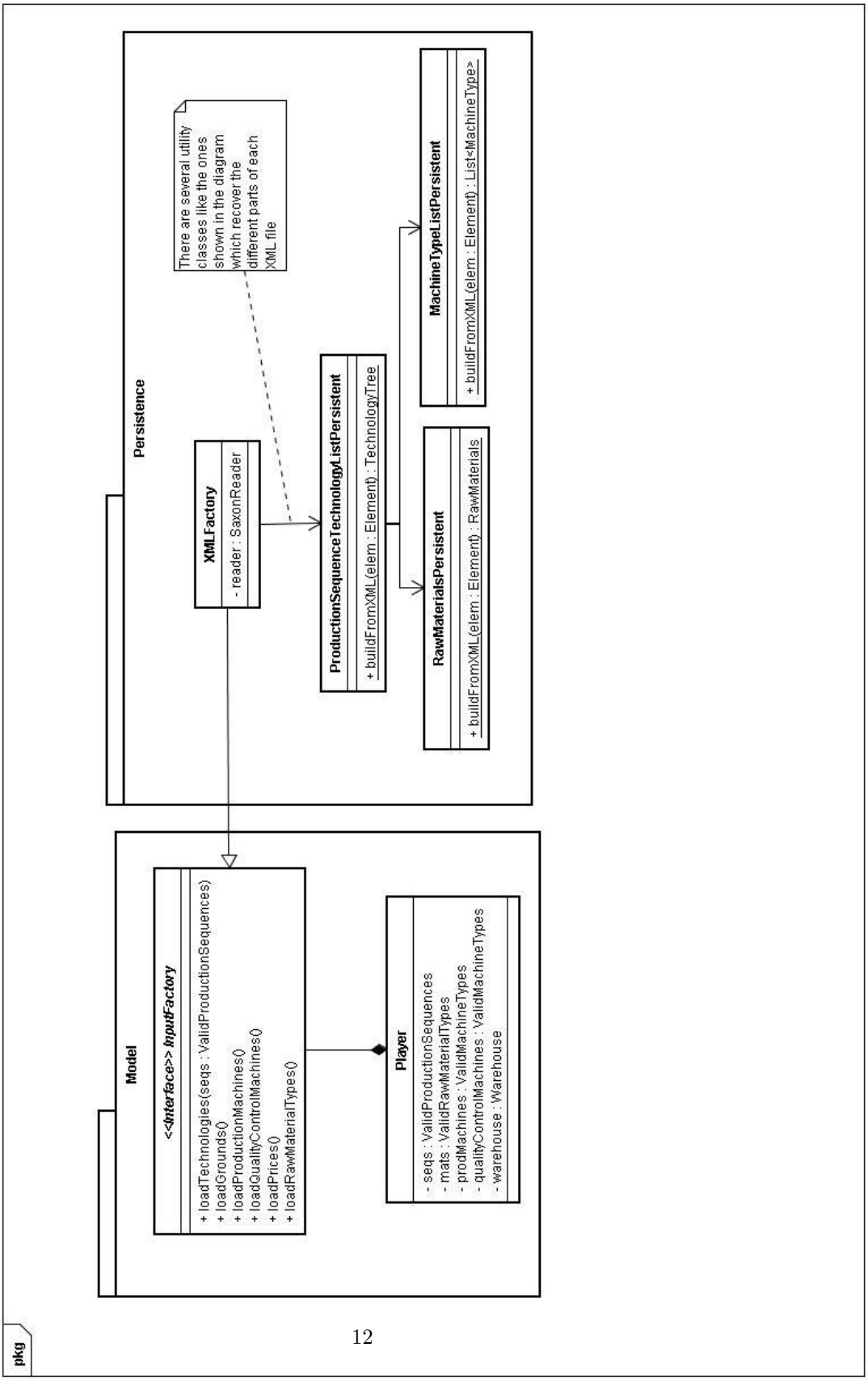

Figura 5: Diagrama de Clases general de la persistencia

En el diagrama de secuencia se puede observar como

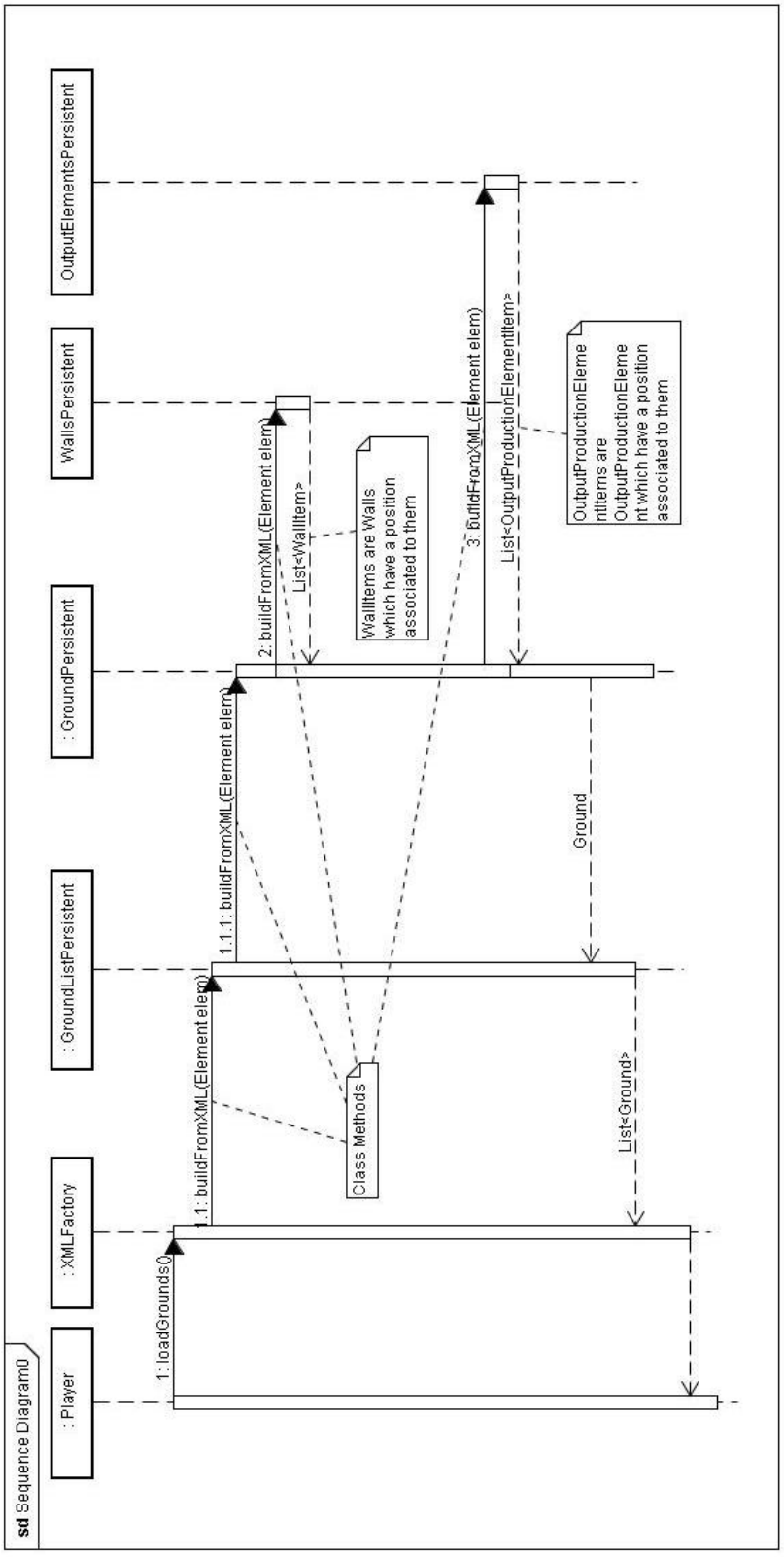

Figura 6: Diagrama de Secuencia de la recuperación de archivos XML

3. Vista de Desarrollo

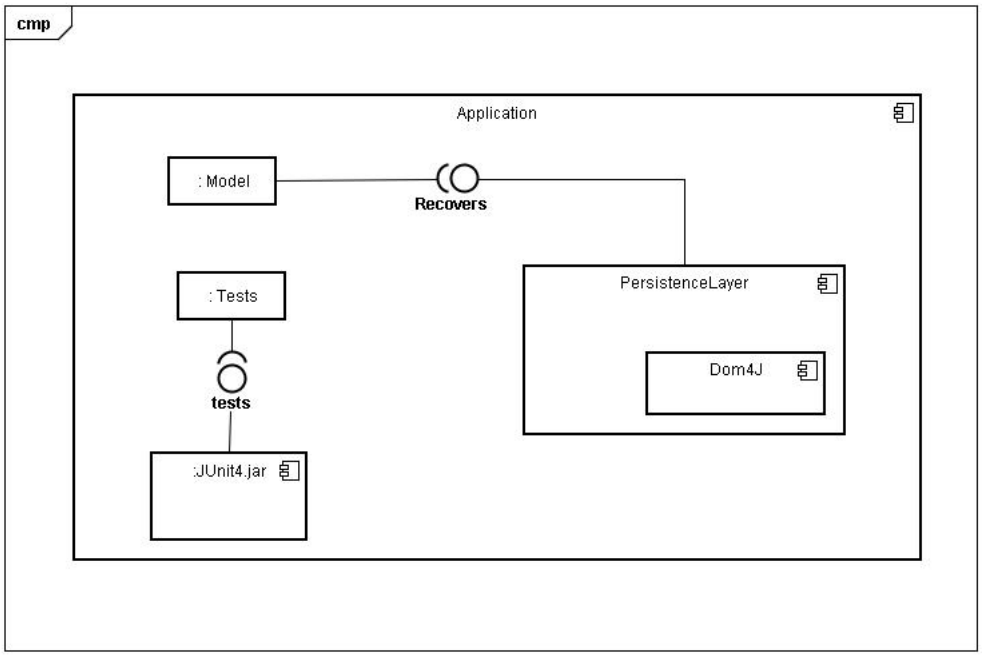

Figura 7: Diagrama de Componentes

En la implementación de la capa de recuperación de datos a la que llamamos persistencia por una posible futura extensión, se utilizó un componente para facilitar la recuperación.

El componente que utilizamos es la libreria Dom4j. La utilización de esta libreria nos permitio desligarnos de la necesidad de hacer un parser de XML.

Otro componente que utilizamos durante el desarrollo fue el framework JUnit. La utilización de este framework facilito la construcción de pruebas que nos sirvieron en primer lugar para ir probando los distintos modulos que se iban implementando y en segundo lugar, para poder refactorizarlos sin miedo a meter nuevas fallas en el programa.

4. Vista de Procesos

La aplicación corre en una computadora, sobre una Java Virtual Machine. Dentro de esta maquina virtual corre un único proceso. No existe comunicación alguna de este proceso con otros procesos.

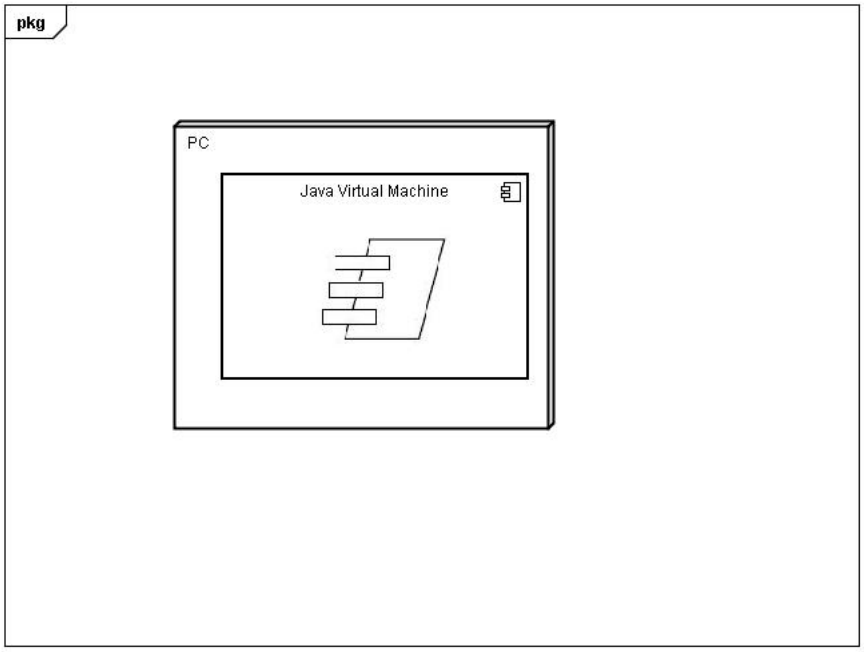

Figura 8: Diagrama de Procesos

En una aplicación como la desarrollada, la complejidad de la vista de procesos es muy simple en comparación al de la vista lógica o de desarrollo.

5. Vista Física

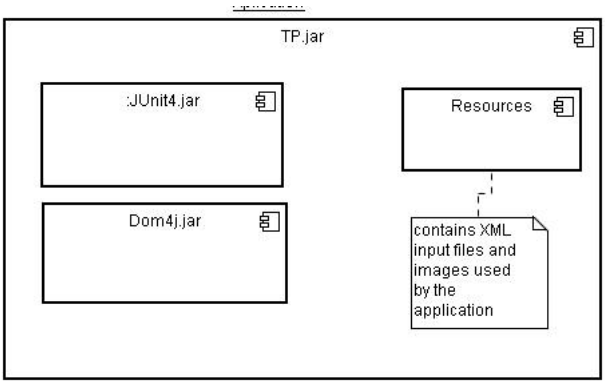

Figura 9: Diagrama de Despliegue

El diagrama de *deployment* es muy simple ya que la aplicación no es una aplicación que este desplegada en más de un sistema. Por lo tanto, todos los componentes existentes estan contenidos en el único nodo del sistema.

Además, se puede observar que el jar de la aplicación contiene dentro de si a los jars de los otros componentes necesarios para el uso del programa. Esto simplifica la instalación del programa ya que no tendra que lidiar con problemas de dependencias. Tiene como desventaja que el usuario puede llegar a tener que tener estas librerias más de una vez en su computadora.

6. Escenarios

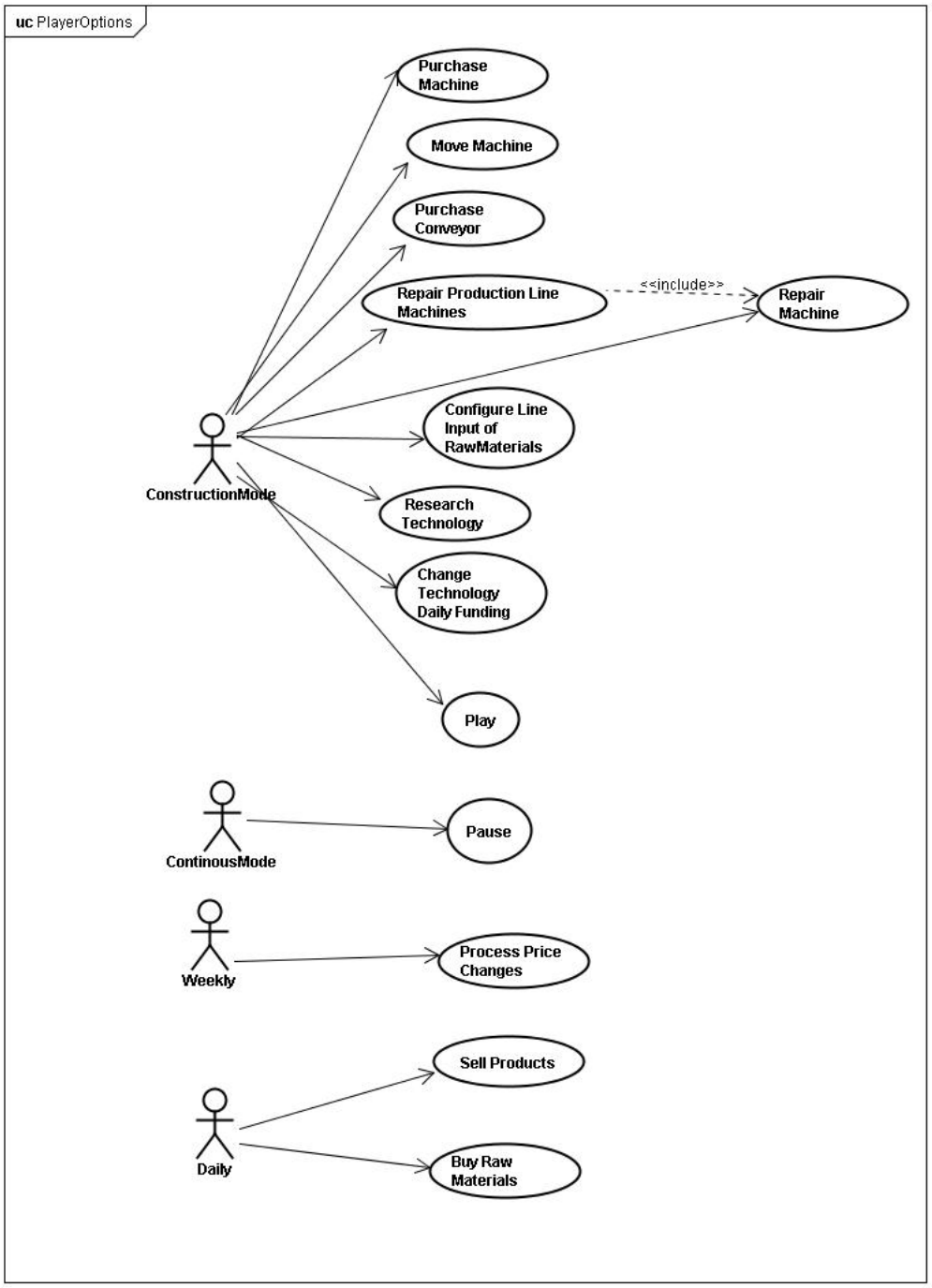

Figura 10: Diagrama de Casos de Uso

6.1. Funcionalidades

1. *Purchase Machine:* El jugador puede comprar máquinas y ubicarlas en la fabrica.

2. *Purchase Conveyor:* El jugador puede colocar segmentos de cintas transportadoras. Una vez conectadas las cintas transportadoras con otros elementos de la línea, la cinta muestra la dirección que seguira el producto. Acción ejecutada desde el modo construcción.

3. *Sell Warehouse:* El jugador puede vender la fabrica y volver a la selección de terrenos.

4. *Sell Machine:* El jugador puede vender la maquinay recibir a cambio un valor proporcional de su precio consecuencia de la depreciación sufrida.

5. *Move Machine:* El jugador puede reubicar máquinas ya compradas en las distintas ubicaciones disponibles de la fabrica. La maquinaqueda ubicada en la fabrica con su respectiva entrada y salida.

6. *Configure Input of Raw Materials:* El jugador selecciona la materia prima que se usara para la producción en la línea.

7. *Repair Production Line Machines:* El jugador selecciona la linea de producción cuyas máquinas desea reparar. La reparación tiene un costo proporcional al precio de la máquina. Acción ejecutada desde el modo construcción.

8. *Repair Machine:* El jugador repara una maquinaque selecciona. La reparación tiene un costo proporcional al precio de la máquina. Acción ejecutada desde el modo construcción.

9. *Play:* El jugador pasa a modo continuo.

10. *Pause:* El jugador pasa a modo construcción. Desde el modo construcción.

11. *Process prices change:* Semanalmente cambian todos los precios de los productos y las materias primas.

12. *Sell Products:* Al final de cada día, la producción es vendida con los precios correspondientes a esa semana.

13. *Buy Raw Materials:* El jugador compra la materia prima que quiera. Acción ejecutada desde el modo construcción.

14. *Research Technology:* El jugador decide cual será la próxima tecnología que sera desarrollada

15. *Change Technology Daily Funding:* El jugador establece cuanta plata invertirá en el desarrollo de nuevas tecnologías.

7. Posibles Mejoras y Extensiones

Una funcionalidad que nos hubiera gustado poder implementar es el hecho de que el jugador pueda solicitar prestamos para que no pierda apenás se queda sin plata.

8. Conclusión

La primer conclusión a la que llegamos fue que en reiteradas ocasiones tomamos decisiones que aumentaban la complejidad del sistema sin hacer un análisis completo de si podíamos llegar a implementarlas a tiempo. Así fue que durante las últimas semanas de desarrollo nos encontramos muy apretados por el tiempo incluso habiendo aprovechado todas las semanas que nos habían provisto para el desarrollo del mismo.

Una segunda conclusión es que es preferible tomarse el tiempo necesario para diseñarlo bien el sistema antes de empezar a escribir código. No hacerlo puede resultar en perder mucho tiempo en caso de que como consecuencia de un requisito que no se había contemplado o una idea que no se había terminado de cerrar, sea necesario replantearse el problema.

Referencias

[1] KRUCHTEN, Philipe. “Architectural Blueprints The *4+1* View Model of Software Architecture”

[2] Wikipedia EN, “4+1 Architectural View Model”