UNIVERSIDAD POLITÉCNICA DE VALENCIA
DEPARTAMENTO DE ORGANIZACIÓN DE EMPRESAS,
ECONOMÍA FINANCIERA Y CONTABILIDAD

PLANIFICACIÓN TÁCTICA DE LAS OPERACIONES EN CADENAS DE SUMINISTRO DE RESPUESTA RÁPIDA (*RESPONSIVE*) CON ESTRUCTURA ALTERNATIVA DE PROCESOS: MODELADO MATEMÁTICO, IMPLEMENTACIÓN Y EXPERIMENTACIÓN

TESIS DOCTORAL

Autor: D. José Luis Calderón Lama

Directores: Dr. José Pedro García Sabater
Dr. Francisco Cruz Lario Esteban

Valencia, Julio de 2011

AGRADECIMIENTOS

Quiero dedicar este pequeño espacio a agradecer sinceramente a todos las siguientes personas e instituciones:

A la Universidad de Piura, que me dio todo el apoyo moral y económico requeridos para hacer el doctorado y llegar a buen fin. Dentro de la Universidad mi agradecimiento específico los Decanos de la Facultad de Ingeniería de los años 2003 a 2011.

Al proyecto de colaboración europea (Red LANMOT), que me dio la subvención económica para estudiar en España durante tres años. Especialmente al Dr. Koenraad Debackere que coordinó dicho proyecto en los años 2002 a 2004 (hasta donde tengo conocimiento).

A la Universidad Politécnica de Valencia (UPV), que me dio la beca de estudios durante tres años y, dentro de la UPV, al Departamento de Organización de Empresas, que me brindó el doctorado. Así mismo a todos los profesores del doctorado por sus clases y guías durante los años 2003 al 2006.

A la Asociación Universitaria Iberoamericana de Postgrado (AUIP), por haberme brindado una beca de movilidad, en el 2010, para viajar a España a hacer la experimentación y redacción final de la tesis durante dos meses.

Al Doctor Francisco Cruz Lario Esteban, sin cuyo personal apoyo no habría venido a la UPV y logrado sobrellevar problemas temporales durante los tres primeros años de mi doctorado. Además por su asesoramiento en mi tesis y la estima personal que me brindó.

Al Doctor José Pedro García Sabater, por haberme dirigido excelentemente bien en mi tesis y por haberme animado y ayudado en momentos difíciles de su ejecución.

Además al señor Ignacio Baños, dueño de la empresa Plastinsa, quien me brindó cortésmente la información necesaria para aplicar mis modelos matemáticos.

También a todos los amigos de la UPV que me ayudaron durante estos tres años a resolver problemas técnicos y que compartieron conmigo ratos de sano esparcimiento.

Pero agradezco especialmente a mi esposa, mis hijos y mis suegros, por su apoyo incondicional y por haber soportado la carga de nuestro viaje a España, nuestra separación temporal y nuestra estrechez económica durante todo el tiempo que llevó realizar este proyecto. Asimismo a mi hermana Kena por haberme apoyado en la corrección de mis capítulos y por su apoyo moral diario.

A aquellos que omito pero que me han brindado su apoyo en distintos aspectos para que logre culminar este programa de doctorado y la tesis doctoral.

A todos, muchas gracias: José Luis Calderón Lama
Julio de 2011

RESUMEN

Se define la Cadena de Suministro de Respuesta Rápida (*Responsive Supply Chain*) como aquella apropiada para productos innovadores con procesos estables. Esta Cadena de Suministro (CS) tiene que enfrentar una incertidumbre alta en la previsión de la demanda de sus múltiples productos (demanda que es además estacional y volátil). Interesa reducir los costes, las roturas de stock y el exceso de productos que deberán rematarse al final de cada temporada de ventas.

El objetivo de esta tesis es mejorar la Planificación Táctica de las Operaciones en las Cadenas de Suministro de Respuesta Rápida con estructura alternativa de procesos (es decir con la posibilidad fabricar los productos de varias maneras aplicando el concepto de aplazamiento -*postponement*-), para productos con ciclo de vida corto (con drástica pérdida de valor en el mercado al final de la temporada de ventas), con proveedores alternativos y con componentes comunes y no comunes. Se busca maximizar los beneficios como diferencia entre los ingresos por ventas y los costes totales de producción, almacenaje y transporte.

En cuanto a la metodología de modelado, se ha tenido en cuenta dos momentos en la toma de decisiones. El primer momento de planificación de las operaciones ocurre varios meses antes del inicio de la temporada de ventas y se planifica contra previsiones muy inciertas de la demanda. El segundo momento de planificación de las operaciones ocurre al inicio de la temporada de ventas, cuando las primeras ventas permiten obtener unas previsiones muy precisas de la demanda. En consecuencia se han desarrollado dos modelos matemáticos de planificación táctica de las operaciones (un modelo estocástico y un modelo determinista) con un nuevo enfoque, el de *strokes*, de manera que se planifican los procesos y no los productos. Con este nuevo enfoque se simplifica el modelado matemático de las múltiples alternativas de aplazamiento y los múltiples procesos de fabricación y transporte.

Para implementar los modelos de manera que se ejecuten consecutivamente, se ha desarrollado una herramienta en Java que: genera un archivo XML, el cual contiene toda la información de entrada al primer modelo, simula la demanda para el segundo modelo (usando el método de difusión de Bass para productos nuevos y la simulación de Monte Carlo), calcula los resultados del primer modelo en el inicio de la temporada de ventas (en función de la demanda simulada) y los ingresa al segundo modelo, ejecuta el segundo modelo tantas veces como el decisor considere convenientes (para igual número de demandas simuladas), calcula los resultados (promedio de las diferentes ejecuciones) y genera otro archivo XML con los resultados. Los modelos matemáticos se han programado en lenguaje AMPL y se resuelven con el optimizador comercial Gurobi Optimizer 3.0.3.

Con la información proporcionada por una empresa valenciana, que produce localmente y además subcontrata (a proveedores en China) la fabricación de componentes y artículos de plástico y metal, se ha experimentado distintas alternativas de costes, incertidumbre de la demanda, momentos de decisión, tiempos de entrega, etc., para distintas problemáticas de la CS.

Se puede afirmar que el conjunto de modelos desarrollados mejora la planificación táctica de las operaciones en CS de respuesta rápida, permite la selección de distintos proveedores alternativos y, permite anticipar y analizar diferentes escenarios de demanda y alternativas de procesos, capacidades, valores de remate, etc.

SUMMARY

The Responsive Supply Chain is defined as the appropriate for innovative products and stable processes. This Supply Chain (SC) has to face a highly uncertain demand forecast for its multiple products (with seasonal and volatile demand). It is important to reduce costs, stock-outs and closings at the end of each selling season.

The objective of this thesis is to improve the Operations Tactical Planning of Responsive Supply Chains with alternative structure of processes (i.e. the possibility to manufacture the products in several ways applying the concept of postponement), for short life cycle products (with drastic loss of value on the market at the end of the sales season), with alternative suppliers, with stable processes and with common and uncommon components. It seeks to maximize the benefits as difference between the income for sales and the total costs of production, storage and transportation.

For the modeling methodology, two moments has been looked in the decision-making. The first moment of tactical operations planning takes place several months before the start of the sales season and is planned to highly uncertain forecasts of demand. The second moment of tactical operations planning occurs at the beginning of the sales season, when the first sales allow obtaining highly accurate forecasts of demand. Therefore two mathematical models of operations tactical planning have been developed (a stochastic and a deterministic model) with a new approach of *strokes*, in which the processes are planned instead of the products. This new approach simplifies mathematical modeling of the multiple alternatives of postponement and multiple processes of manufacture and transportation.

In order to implement the consecutive execution of the models, a tool in Java has been developed that: generates a XML file which contains all the data entry to the first model, simulates demand for the second model (using the Bass diffusion method for new products and the simulation of Monte Carlo), calculates the results of the first model in the beginning of the sales season (based on the simulated demand) and enters them in the second model, executes the second model as many times as the decider considers suitable (for equal number of simulated demands), calculates results (average of different executions) and generates antocher XML file with the results. The mathematical models have been programmed in AMPL language and they are solved with the commercial optimizer Gurobi Optimizer 3.0.3.

Based on the information provided by a Valencian company, that manufactures locally and in addition subcontracts (to a suppliers in China) the manufacture of components and articles of plastic and metal, different alternatives of costs have been experimented and also different: demand uncertainty, moments for the execution of the second model, lead times, etc., for different problematic of the SC.

It is possible to affirm that the set of developed models improves the responsive SC operations tactical planning, allows the selection of different alternative suppliers and, allows to anticipate and to analyze different demand scenes and alternatives of processes, capacities, values of clearance sale, etc.

RESUM

Es definix la Cadena de Subministrament de Resposta Ràpida (*Responsive Supply Chain*) com aquella apropiada per a productes innovadors amb processos estables. Esta Cadena de Subministrament (CS) ha d'enfrontar una incertesa alta en la previsió de la demanda dels seus múltiples productes (demanda que és a més estacional i volátil). Interessa reduir els costos, les ruptures d'estoc i l'excés de productes que hauran de rematar-se al final de cada temporada de vendes.

L'objectiu d'esta tesi és millorar la Planificació Tàctica de les Operacions en les Cadenes de Subministrament de Resposta Ràpida amb estructura alternativa de processos (es dir amb la possibilitat fabricar els productes de diverses maneres aplicant el concepte d'ajornament -*postponement*-), per a productes amb cicle de vida curt (con dràstica pèrdua de valor en el mercat al final de la temporada de ventas), amb proveïdors alternatius i amb components comuns i no comuns. Es busca maximitzar els beneficis com a diferència entre els ingressos per vendes i els costos totals de producció, magatzematge i transport.

Quant a la metodologia de modelatge, s'ha tingut en compte dos moments en la presa de decisions. El primer moment de planificació de les operacions ocorre uns quants mesos abans de l'inici de la temporada de vendes i es planifica contra previsions molt incertes de la demanda. El segon moment de planificació de les operacions ocorre a l'inici de la temporada de vendes, quan les primeres vendes permeten obtindre unes previsions molt precises de la demanda. En conseqüència s'han desenrotllat dos models matemàtics de planificació tàctica de les operacions (un model estocástico i un model determinista) amb un nou enfocament, el de *strokes*, de manera que es planifiquen els processos i no els productes. Amb este nou enfocament se simplifica el modelatge matemàtic de les múltiples alternatives d'ajornament i els múltiples processos de fabricació i transport.

Per a implementar els models de manera que s'executen consecutivament, s'ha desenrotllat una ferramenta a Java que: genera un arxiu XML, el qual conté tota la informació d'entrada al primer model, simula la demanda per al segon model (usando el mètode de difusió de Bass per a productes nous i la simulació de Monte Carlo), calcula els resultats del primer model en l'inici de la temporada de vendes (en funció de la demanda simulada) i els ingressa el segon model, executa el segon model tantes vegades com el decisor considere convenients (para el mateix nombre de demandes simuladas), calcula els resultats (promedio de les diferents ejecuciones) i genera un altre arxiu XML amb els resultats. Els models matemàtics s'han programat en llenguatge AMPL i es resolen amb l'optimitzador comercial Gurobi Optimizer 3.0.3.

Amb la informació proporcionada per una empresa valenciana, que produïx localment i a més subcontracta (a proveïdors en China) la fabricació de components i articles de plàstic i metall, s'ha experimentat distintes alternatives de costos, incertesa de la demanda, moments de decisió, temps d'entrega, etc., per a distintes problemátic de la CS.

Es pot afirmar que el conjunt de models desenrotllats millora la planificació tàctica de les operacions en CS de resposta ràpida, permet la selecció de distints proveïdors alternatius i, permet anticipar i analitzar diferents escenaris de demanda i alternatives de processos, capacitats, valors d'acabament, etc.

Planificación Táctica de las Operaciones en Cadenas de Suministro de Respuesta Rápida (*Responsive*) con Estructura Alternativa de Procesos: Modelado Matemático, Implementación y Experimentación

Índice

Índice de figuras

Índice de tablas

Planificación Táctica de las Operaciones en Cadenas de Suministro de Respuesta Rápida (*Responsive*) con Estructura Alternativa de Procesos: Modelado Matemático, Implementación y Experimentación

1. INTRODUCCIÓN

1.1. Presentación

Actualmente, las Cadenas de Suministro (CS) abarcan empresas en distintos países, que planifican y coordinan sus capacidades (de producción, almacenamiento, transporte, etc.) para brindar al mercado productos en el tiempo, cantidad, calidad, variedad y coste adecuados. Para lograrlo, deben además planificar la producción abarcando dos o más niveles (montaje final, submontaje, fabricación de componentes, etc.) en varias empresas de la cadena de suministro, con distintos plazos de entrega y costes.

En la década de los 90s, algunos investigadores indicaron la necesidad de hacer coincidir el tipo de Cadena de Suministro (CS) con las características del producto y los requerimientos del mercado (Hoekstra y Romme (1992), Fuller et al. (1993), Olhager (1994), Berry et al. (1995), Fisher (1997), Copacino (1997), Gattorna (1998)).

Los productos se pueden clasificar según, entre otros factores, la predictibilidad de su demanda, la variedad, el margen de contribución y la duración de su ciclo de vida. Una clasificación útil es la propuesta por Fisher (1997), quien los cataloga en funcionales e innovadores. Los productos funcionales tienen demanda predecible, baja variedad, margen de contribución bajo y ciclo de vida largo, mientras que los productos innovadores tienen demanda impredecible, alta variedad, margen de

contribución alto y ciclo de vida corto. Fisher (1997) además afirma que los productos funcionales deben tener una CS Eficiente, mientras los productos innovadores deben tener una CS de Respuesta Rápida. La CS Eficiente debe satisfacer la demanda predecible al menor coste posible, mientras la CS de Respuesta Rápida debe responder rápido a la demanda impredecible minimizando roturas de stock, rebajas de fin de temporada e inventario obsoleto, por ello esta última CS debe usar diseño modular para aplazar la diferenciación del producto lo más posible.

Las CS dependen también de la incertidumbre del aprovisionamiento (Lee, 2002). Los procesos estables tienen incertidumbre baja en el proceso de aprovisionamiento y los procesos en evolución la tienen alta. La CS de Respuesta Rápida (para productos innovadores) requiere de procesos estables (Lee, 2002). Esta CS debería utilizar proveedores confiables y coordinar con ellos las cantidades a producir de cada componente, sub-ensamble y producto final, anticipándose a la demanda real, de manera que satisfagan el mercado con el mayor nivel de servicio y el mayor beneficio posible.

El cambio de política industrial y comercial de muchos países sub-desarrollados, ha facilitado que los países desarrollados subcontrataran la producción de muchos bienes a los países de bajos costes de mano de obra (Stratton y Warburton, 2003). Esto ha obligado a una mayor anticipación en la planificación de la producción puesto que los tiempos de aprovisionamiento de proveedores de un país lejano son mayores que los de los proveedores locales.

Para responder rápidamente a la demanda de productos innovadores (impredecible) minimizando roturas de stock, rebajas de fin de temporada e inventario obsoleto, Wong et al. (2006) han propuesto la CS *market responsive* (sensible al mercado), la cual considera tanto proveedores lejanos como proveedores locales y abarca una CS híbrida con estrategias de producción MTS (*Make to Stock*) y ATO (*Assemble to Order*). Según su propuesta, se debe anticipar la producción de parte de la demanda y lanzar al mercado estos productos antes del inicio de la temporada de ventas, para luego, con una previsión más precisa de la demanda, ensamblar los productos finales a partir de un stock de componentes y hacer los envíos subsiguientes al mercado. Esto requiere una mayor planificación y coordinación de las principales empresas de la CS.

La planificación de la Cadena de Suministro se da en tres niveles: Estratégico, Táctico y Operativo (Shapiro (1998), Chopra y Meindl (2001), Van Landeghem y Vanmaele (2002)). Rohde et al. (2000), Fleischmann et al. (2005) y Stadtler (2005) clasifican las tareas de planificación, de acuerdo al tiempo que abarcan y a la importancia de las decisiones, en: planificación de largo plazo, planificación de medio plazo y planificación de corto plazo. Afirman que en la planificación de la producción a medio plazo se hace el Plan Maestro de Producción (PMP) y la Planificación de la Capacidad Aproximada (PCA).

La presente tesis abarca la planificación táctica de las operaciones de cadenas de suministro de productos innovadores con procesos estables (CS de tipo *Responsive*), con proveedores conocidos, con ciclo de vida corto y con la posibilidad de hacer aplazamiento de la ejecución de algunas operaciones.

La tesis está estructurada de este modo: en el capítulo 1 se encuentra la Introducción, el Objetivo de la tesis y el Esquema general. En el capítulo 2 se presenta el Estado del Arte. En el capítulo 3 se describe el problema general, se hace el modelado, se diseñan los modelos matemáticos y se formulan los algoritmos necesarios para su implementación. En el capítulo 4 se hace la experimentación y se analizan los resultados. Por último, en el capítulo 5 se presentan las conclusiones y las líneas futuras de investigación.

1.2. Objetivo y alcance de la tesis

A partir de la experiencia que ha ido acumulando el Centro de Investigación, Gestión e Ingeniería de Producción (CIGIP) y el Grupo ROGLE a través de diversos Proyectos Europeos y convenios de colaboración, se plantea como objetivo para el presente trabajo: el análisis del problema de la Planificación Táctica (Planificación a medio plazo) de las Operaciones para Productos Innovadores con Procesos Estables en Cadenas de Suministro con Estructura Alternativa de Procesos[1].

[1] Estructura Alternativa de Procesos: son varias alternativas de producción de un mismo producto; de manera que se puede fabricar de una sola vez de materia prima a producto terminado (sin almacenamiento intermedio), y también se pueden fabricar componentes, almacenándolos hasta definir oportunidad y cantidad de hacer subensambles, y almacenarlos hasta concretar momento y cantidad de montar ensambles finales; o también fabricar sólo los componentes comunes y esperar el momento adecuado para producir los componentes no comunes y hacer el subensamble y ensamble final juntos. En cada caso se pueden emplear diferentes tecnologías de producción.

Este estudio abarca el caso general de Cadenas de Suministro integradas por empresas que diseñan productos de temporada y subcontratan la fabricación y montaje a distintos proveedores tanto nacionales como extranjeros (en países cuyo coste de producción es bajo), es decir abarca las etapas de fabricación desde materias primas hasta productos finales y el transporte de las plantas lejanas a las plantas locales (no abarca la distribución y venta detallista). El número y localización de los posibles proveedores es conocido. Los productos tienen demanda estacional con un pico alto en sólo dos o tres meses al año, luego del cual los productos pierden valor de mercado (teniendo un precio de remate bajo y conocido pero que puede estar incluso debajo del coste). El precio de los productos terminados es conocido y fijo hasta que acaba el pico de demanda, momento en el que los productos tienen un cierto valor residual que es el precio de remate. En la planificación táctica de las operaciones se busca maximizar los beneficios como diferencia entre los ingresos por ventas (al precio de venta y al valor de residual) y los costes totales (de producción, inventario y transporte).

Por lo anterior, la empresa debe planificar la fabricación de sus productos en función de previsiones de demanda de los artículos finales (sólo éstos tienen demanda externa), considerando dos o más niveles de componentes (lista de materiales) que pueden ser compartidos o no entre los diferentes productos, y abarcando toda una temporada de ventas y los meses previos necesarios para la producción.

La planificación debe empezar con previsiones de la demanda con varios meses de anticipación con respecto al inicio de la temporada de ventas, y en función de estas previsiones se debe contratar la capacidad necesaria en los proveedores extranjeros (considerando además que el tiempo de envío de los proveedores lejanos es mayor que el plazo de entrega del proveedor local). También se debe considerar que la capacidad de los proveedores es finita pero puede ser distinta para cada periodo de tiempo (con límites impuestos por cada sub-contratista) y que los costes de las materias primas y su transporte son diferentes para cada proveedor. Luego, se re-planifica la producción al inicio de la temporada de ventas o cuando se tenga una mejor previsión de la demanda (este nuevo plan debe respetar las capacidades contratadas y costear, si hubiera, costes por sobre-utilización o sub-utilización de la capacidad disponible). Finalmente se debe considerar que se pueda aplazar algunas operaciones

(*postponement*), pudiendo cada fabricante de la CS producir cada producto totalmente o producir algunas de sus partes y sus subensambles, almacenarlos, y después realizar el ensamble final o enviarlos a otro fabricante de la CS.

En consecuencia, para cumplir el objetivo de la presente investigación, que es fundamentalmente mejorar la Planificación Táctica de las Operaciones de CS con las características expuestas en los párrafos anteriores, se tiene como objetivo conjunto desarrollar modelos matemáticos, empleando distintas herramientas (Programación Lineal, Programación Lineal-Entera Mixta, Programación Estocástica, Simulación, etc.), y seleccionar, coordinar y poner a punto las herramientas que se consideren más apropiadas. Dichos modelos tienen que ser particularmente adecuados para determinar en qué condiciones conviene contratar la producción a proveedores cercanos frente a proveedores lejanos o a ambos en distintos periodos y cantidades, y además determinar cómo afectan los resultados: distintos grados de incertidumbre de la demanda, distintos valores residuales del inventario final, distintos costes de sub-utilización, distintos momentos de re-planificación, distintas capacidades de los procesos y distintos costes de pedidos pendientes.

1.3. Esquema general de la tesis

La tesis muestra en primer lugar la Presentación y establece los Objetivos y el Alcance de la misma.

En segundo lugar, se realiza un Estado del Arte que comprende las estrategias de las CS de productos innovadores y los modelos de planificación táctica de la producción (publicados en distintas revistas y actas de congresos) que abarcan dos o más etapas de la cadena de suministro (proveedores, fabricantes, etc.) pero sin entrar en los detalles de la planificación operativa (de corto plazo).

A continuación, en función de las características del problema abarcado, se desarrollan dos modelos, uno estocástico y otro determinista, correspondientes a los dos momentos de planificación (primero el modelo estocástico con varios escenarios y su probabilidad asociada para el primer momento de decisión, luego el modelo determinista para el segundo momento de decisión). Estos modelos permiten el aplazamiento de uno o más procesos, y consideran las condiciones de los proveedores

(pedidos en firme con varios periodos de anticipación y límites de variación de la capacidad en distintos periodos).

En cuarto lugar, se diseña una herramienta en Java para la implementación de los dos modelos en forma consecutiva. Esta herramienta comprende cinco algoritmos para ingresar los datos, para pasar los resultados del primer modelo al segundo modelo, para simular la demanda de la segunda etapa, para ejecutar el segundo modelo tantas veces como desee el decisor y, finalmente para sacar promedios de las ocurrencias y escribir los resultados en hojas de Excel.

A continuación, se ejecuta el modelo completo (que incluye los dos modelos y los algoritmos) con costes y plazos extraídos de una empresa real, se obtienen las soluciones óptimas de diferentes casos y se realizan varios análisis de sensibilidad. Con los resultados obtenidos se elaboran tablas para mostrar la variación de los costes totales, del nivel de servicio, de las ventas y de los beneficios en función de: diferentes costes unitarios, varios grados de incertidumbre de la demanda, distintas capacidades, diferentes costes de sobre y sub utilización de la capacidad, varios momentos de re-planificación y distintos valores residuales.

Finalmente se presentan las Conclusiones y las Líneas Futuras de Investigación.

1.4. Referencias

Berry, D., et al. (1995). Business process for reengineering an electronics products supply chain. IEE Proc. Sci. Meas. Tech., Vol. 142, Nº 5, pp. 395-403.

Copacino, W.C. (1997). Supply Chain Management: The Basics and Beyond. The St. Lucie Press/APICS series on Resource Mangement. Publisher: CRC Press LLC.

Chopra, S. y Meindl, P. (2001). Supply Chain Management: Strategy, Planning and Operations. 1st ed. Prentice Hall.

Fisher, M. (1997). What is the right supply chain for your product? *Harvard Business Review*, 75(2), 105-117.

Fleischmann, B.; Meyr, H. y Wagner, M. (2005). Advanced Planning. En Stadtler, H. y Kilger, C. eds. Supply Chain Management and Advanced Planning. Springer. pp. 81-106.

Fuller, J. B., O'Connor, J. y Rawlinson, R. (1993). Tailored logistics: the next advantage. *Harvard Business Review*, 71 (3), 91-3, pp. 87-98.

Gattorna, J. (ed.) (1998). Strategic Supply Chain Alignment: Best Practice in Supply Chain Management. Gower Publishing Co., Hampshire, U.K.

Hoekstra, S. y Romme, J. (1992). Integrated Logistics Structures: Developing Customer Oriented Goods Flow. McGraw-Hill, London, U.K.

Olhager, J., 1994. On the positioning of the customer order decoupling point. Proceedings of the Pacific Conference on Manufacturing, Jakarta, pp. 1093–1100.

Lee, H.L. (2002). Aligning Supply Chain Strategies with Product Uncertainties. *California Management Review,* 44(3), Spring, 105-119.

Rohde, J., Meyr, H. y Wagner, M. (2000). Die Supply Chain Planning Matrix. *PPS-Management*, 5(1), 10-15.

Shapiro, J. (1998). Bottom-up versus top-down approaches to supply chain modelling. En: Quantitative Models for Supply Chain Management. Kluwer Academic Publishers, Dordrecht, pp. 739–759.

Stadtler, H. (2005). Supply chain management and advanced planning—basics, overview and challenges. *European Journal of Operational Research*, 163(3), 575–588.

Stratton, R. y Warburton, R.D.H. (2003). The strategic integration of agile and lean supply. *International Journal of Production Economics*, 85(2), 183–198

Van Landeghem, H. y Vanmaele, H. (2002). Robust planning: a new paradigm for demand chain planning. *Journal of Operations Management*, 20(6), 769–783.

Wong, C.Y., Arlbjørn, J.S., Hvolby, H-H. y Johansen, J. (2006). Assessing responsiveness of a volatile and seasonal supply chain: A case study. *International Journal of Production Economics,* 104(2), 709-721.

2. ESTADO DEL ARTE

2.1. Introducción

La gran interrogante de esta tesis es ¿cómo mejorar la planificación táctica de las operaciones de las cadenas de suministro de respuesta rápida *(responsive)* con estructura alternativa de procesos? Y para responder esa gran pregunta se tienen que responder unas preguntas intermedias que pueden ser:

1. ¿Existen actualmente diferentes estrategias de las CS?
2. ¿En qué circunstancias se debe emplear cada una?
3. ¿Cómo se realiza actualmente la planificación táctica de las operaciones de CS de respuesta rápida sin/con estructura alternativa de procesos?
4. ¿Cómo mejorar la planificación táctica de las operaciones de las cadenas de suministro de respuesta rápida (*responsive*) con estructura alternativa de procesos?

Las tres primeras preguntas se abordarán en el presente capítulo, y la cuarta pregunta se contestará en el capítulo 4: Aplicación de los modelos y Análisis.

El resto de este capítulo se organiza como sigue, en el segundo apartado se presentan las estrategias de las cadenas de suministro; en el tercer apartado se determinan los requisitos para la planificación de la producción en cadenas de suministro de productos innovadores, en el cuarto apartado se muestran los modelos matemáticos que abarcan a la planificación de la producción de medio plazo y en el quinto apartado se revisa la programación estocástica. Las conclusiones se incluyen en el sexto apartado de este capítulo.

2.2. Estrategias de Cadenas de Suministro

El objetivo de este apartado es encontrar las respuestas a las preguntas:

- ¿Existen actualmente diferentes estrategias de las CS?
- ¿En qué circunstancias se debe emplear cada una?

El medio para lograrlo es buscar y analizar lo que se haya escrito hasta la fecha sobre estrategias de las cadenas de suministro, en especial de cadenas de suministro de respuesta rápida (*responsive*), en libros, revistas de investigación y congresos.

A continuación se presentarán los aportes encontrados de distintos investigadores, en orden cronológico, luego se hará una tabla comparativa y un resumen, y finalmente se concluirá con las respuestas buscadas.

Hoekstra y Romme (1992), afirman que la ubicación del punto de desacople en la CS está en función del grado de personalización del producto, y que el punto de desacople coincide con el punto de penetración de la orden. En consecuencia argumentan que el punto de desacople debe ser el lugar principal de stock. El punto de desacople indica el eslabón de la cadena de suministro hasta el cual la estrategia de la CS es tipo *push* (trabajar contra previsiones de la demanda) y a partir del cual la estrategia de la CS pasa a ser tipo *pull* (trabajar contra pedido). En la figura 2.1 se muestra el punto de desacople como punto principal de stock (indicado por los triángulos) en función de cuatro grados de personalización:

- *Buy to order:* productos diseñados y fabricados en función de los Pedidos de los usuarios finales; en este caso los Inventarios los tienen los Proveedores de materias primas (*Raw Material Supplier*).
- *Make to order:* productos fabricados sobre Pedido en base a un catálogo; en este caso el stock de materias primas se ubica en el fabricante (*Manufacturers*).
- *Assemble to order:* productos ensamblados sobre Pedido, cuyo principal stock es de componentes y subensambles y se ubica en ensamblador (*Assemblers*).
- *Ship to stock:* productos producidos contra Previsiones, con inventarios de productos terminados en el distribuidor y en el detallista (*Retailer*).

Además, la figura 2.1 muestra el efecto que se desea obtener al fijar un punto de desacople sobre el comportamiento de la demanda. Aguas abajo del punto de desacople, la demanda es muy variable, mientras aguas arriba de dicho punto la visión de la demanda es más uniforme.

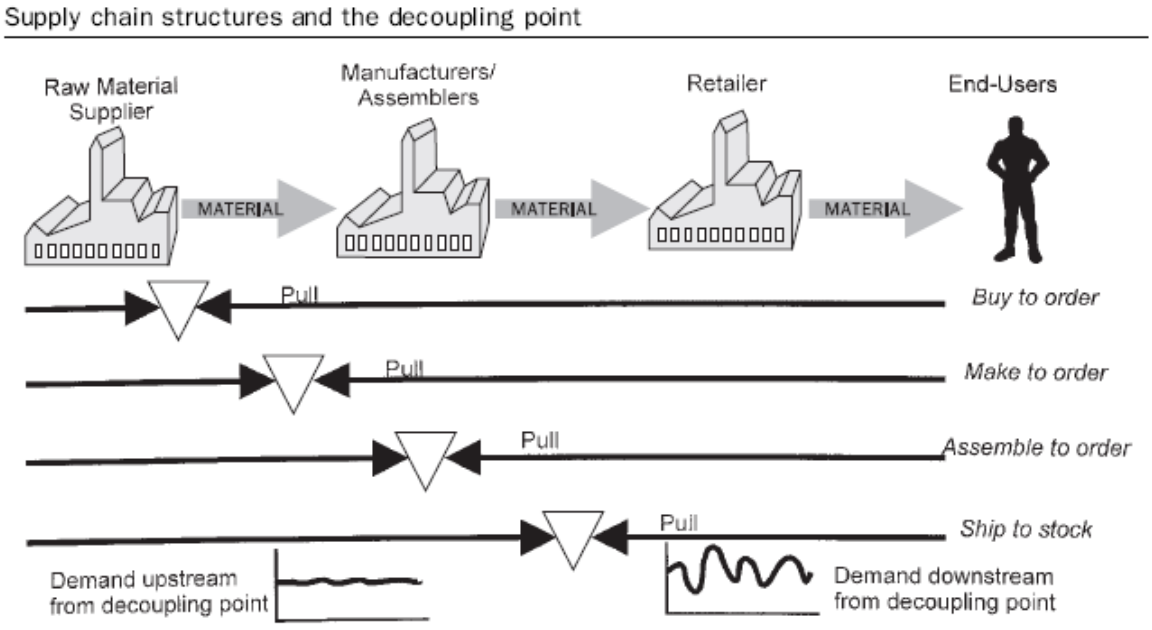


Figura 2.1: Cadenas de suministro según la ubicación del punto de desacople
(Hoekstra y Romme, 1992)

Olhager (1994) establece la relación entre el *grado de personalización del producto* y la Estrategia de Producción (*MTS, ATO, MTO, ETO*) (ver la tabla 2.1).

Grado de personalización	Estrategia de Producción	Ubicación del punto de desacople y del punto de penetración de la orden
Bajo	MTS	En el punto final de venta
Medio	ATO	En el centro de montaje
Alto	MTO	En el centro de fabricación de partes y componentes
Muy alto	ETO	En el centro de diseño

MTS: Make to Stock. MTO: Make to Order.
ATO: Assemble to Order. ETO: Engineer to Order.

Tabla 2.1: Grado de personalización del producto y
Estrategia de Producción (adaptado de Olhager, 1994)

Fisher (1997) afirma que los productos con demanda predecible y ciclo de vida largo (productos funcionales) deben tener una CS Eficiente, mientras los productos con demanda impredecible y ciclo de vida corto (productos innovadores) deben tener una CS de Respuesta Rápida. La CS Eficiente debe maximizar el desempeño y

minimizar el coste, mientras la CS de Respuesta Rápida debe lograr la disponibilidad de los productos en el momento y cantidad requeridos y a bajo coste, por ello debe tener el punto de desacople entre la fabricación de componentes y el ensamble final para aplazar la diferenciación del producto lo más posible (ver la tabla 2.2).

	CS Eficientes	**CS de Respuesta Rápida**
Propósito primario	Satisfacer la demanda predecible eficientemente al menor coste posible	Responder rápido a la demanda impredecible pero minimizando roturas de stock (*stockouts*), rebajas de fin de temporada e inventario obsoleto
Enfoque de manufactura	Mantener un alto promedio de tasa de utilización	Disponer de capacidad en exceso
Estrategia de Inventarios	Generar alta rotación y minimizar el inventario a lo largo de la cadena	Disponer de significativos stocks de partes y bienes terminados
Enfoque de tiempo total de entrega (*lead time*)	Acortar los tiempos hasta donde se pueda sin que incrementen el coste	Invertir agresivamente en maneras de reducir el tiempo total de entrega
Criterio para elegir proveedores	Elegir principalmente por coste y calidad	Elegir principalmente por velocidad, flexibilidad y calidad
Estrategia de producto-diseño	Maximizar el desempeño y minimizar el coste	Usar diseño modular para aplazar la diferenciación del producto lo más posible

Tabla 2.2: Cadenas Eficientes versus de Respuesta Rápida (Fisher, 1997)

Naylor et al. (1999) y Mason-Jones et al. (2000) analizan la relación entre las cadenas de suministro y las "prioridades de los consumidores" (*customer drivers* en el original). Identifican tres tipos de CS en función de los ganadores de mercado (*market winners*) y calificadores del mercado (*market qualifiers*) de los productos:

ajustada (*lean* en el texto original), ágil (*agile*) y ágil-ajustada (*leagile*) (ver la figura 2.2).

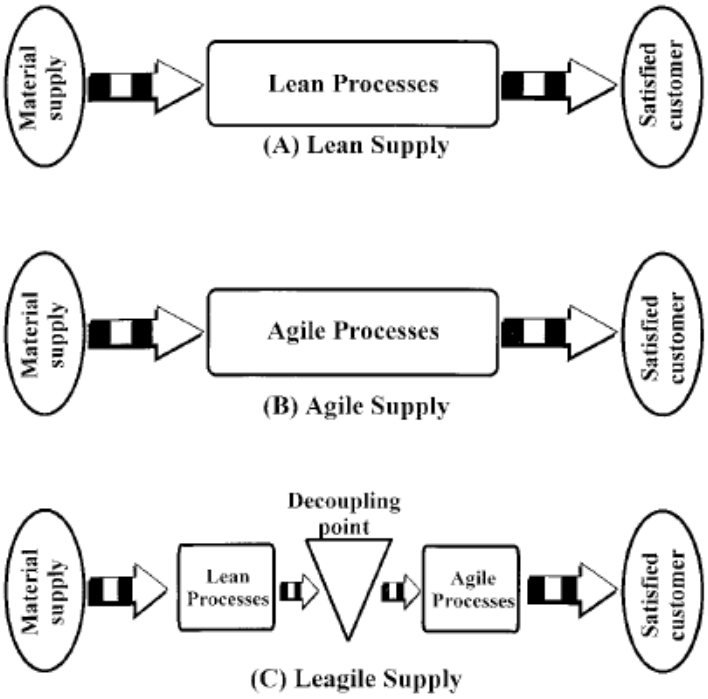


Figura 2.2: Ubicación del punto de desacople para la CS leagile
(Mason-Jones, Naylor, and Towill 2000)

- CS ajustada (*lean*): procura eliminar todo desperdicio, incluso de tiempo, y asegurar una producción nivelada para obtener el menor coste posible a lo largo de toda la CS. Esta CS es adecuada para productos cuyo ganador de mercado es el coste.

- CS ágil (*agile*): usa el conocimiento del mercado y las asociaciones virtuales entre empresas para explotar oportunidades rentables en mercados volátiles. Esta CS es adecuada para productos cuyo ganador de mercado es la flexibilidad.

- CS "agiltada" (*leagile*): es la combinación de los paradigmas lean y agile en la estrategia de la cadena de suministro; consiste en ubicar el punto de desacople donde sea mejor para atender una demanda volátil aguas abajo y tener producción nivelada aguas arriba. La CS "agiltada" es la más apropiada cuando el ganador de mercado es la disponibilidad seguida cercanamente por el coste.

Lamming et al. (2000) introducen la consideración de la complejidad del producto (la complejidad está en función del número de componentes y el número de niveles en las listas de materiales de los productos finales) y, basándose en el trabajo de Fisher (1997), distinguen cuatro tipos de productos. Clasifican la complejidad de los productos en alta o baja e identifican las prioridades competitivas del correspondiente sector industrial de cada tipo de producto:

- *Productos innovadores de alta complejidad*: sus prioridades competitivas son la flexibilidad, tiempos de entrega cortos, innovación y calidad.
- *Productos innovadores de baja complejidad*: sus prioridades competitivas son tiempos de entrega cortos, flexibilidad, innovación y calidad.
- *Productos funcionales de alta complejidad*: cuyas prioridades competitivas son la minimización de costes, calidad sostenible y el servicio.
- *Productos funcionales de baja complejidad*: cuyas prioridades competitivas son costes y servicio (bajo plazo de entrega y alta confiabilidad).

Christopher y Towill (2001) proponen tres estrategias híbridas que combinan los sistemas *lean* y *agile.* Estas tres estrategias son complementarias en vez de mutuamente exclusivas, sin embargo es probable que cada una trabaje mejor en ciertas condiciones. En la tabla 2.3 (ver en la siguiente página) resumen un conjunto de condiciones apropiadas para la aplicación de estas tres estrategias híbridas.

La estrategia Base e Imprevista (*Base* y *Surge*) consiste en separar la previsión de la demanda en dos partes. La parte base (*Base*) es la parte segura de la Demanda y se puede emplear para comprometer la Producción con bastante anticipación a la temporada de ventas; esta producción se puede realizar en una CS ajustada (*lean*) para lograr economías de costes. La parte imprevista (*Surge*) es muy difícil de prever y requiere esperar hasta estar más próximo de la temporada de ventas para tener una buena Previsión; esta parte de la demanda debe ser satisfecha por procesos más flexibles y de menor plazo de entrega como los de una CS ágil (*agile*). La estrategia *base* y *surge* está siendo empleada cada vez más en la industria de la moda donde la demanda base puede ser satisfecha por países de bajo coste y la demanda *surge* puede ser satisfecha localmente (cerca del mercado).

Estrategias híbridas	Condiciones de mercado apropiadas y ambiente de operación
Pareto 80/20 Usa métodos *lean* para las líneas de alto volumen y métodos *agile* para los de baja movimiento.	Alta variedad; demanda alta para algunos productos y baja para otros.
Punto de desacople Es *lean* aguas arriba del punto de desacople y *agile* aguas abajo del mismo.	Producción *modular* posible e *Inventario intermedio*; aplaza la configuración final o la distribución.
Separación de la demanda en "surge" y "base" Gestiona la parte previsible de la demanda usando principios *lean* y usa principios *agile* para la parte menos previsible de la demanda.	Cuando una parte de la demanda puede ser prevista con seguridad a partir de la experiencia pasada y cuando se dispone de capacidad de producción local.

Tabla 2.3: Estrategias híbridas y condiciones apropiadas para su operación
(Christopher y Towill, 2001)

Lee (2002) vuelve sobre las ideas de Fisher (1997) y examina la parte de suministro de la CS. Señala que las incertidumbres en torno al suministro tienen la misma importancia que las características de la demanda para establecer la estrategia correcta de la CS. Define que un proceso de suministro "estable" es aquel cuyo proceso de manufactura y su tecnología son maduros y la base de aprovisionamiento está bien establecida. En cambio, un proceso de suministro "en evolución" es aquel cuyo proceso de manufactura y tecnología de base está aún en desarrollo inicial y está cambiando rápidamente, y en consecuencia la base de suministro puede ser limitada en tamaño y experiencia. En un proceso de suministro estable, la complejidad de la manufactura suele ser baja o manejable. Los procesos de manufactura estables tienden a ser altamente automatizados y prevalecen los contratos de suministro de largo plazo. En un proceso de suministro en evolución el proceso de manufactura requiere muchos ajustes (*fine-tunning* en el original) y frecuentemente ocurren averías y variaciones en la producción. La base de aprovisionamiento puede no ser muy confiable en la medida que los proveedores están innovando sus procesos. Lee (2002) propone cuatro tipos de la CS (ver la figura 2.3) en función de la incertidumbre en la demanda y de la incertidumbre en el aprovisionamiento.

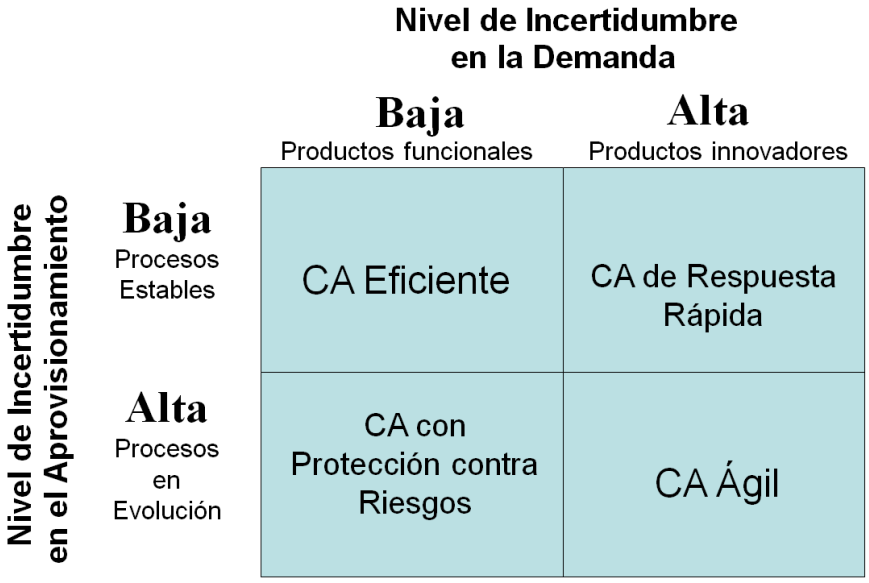


Figura 2.3: Tipos de CS según la incertidumbre en la demanda y en el aprovisionamiento (Lee, Hau L., 2002)

- *CS Eficientes*: utilizan estrategias destinadas a crear la mayor eficiencia de costes en la CS.
- *CS de Respuesta Rápida*: utiliza estrategias dirigidas a ser sensible y flexible a los cambios y distintas necesidades de los clientes.
- *CS con Protección contra Riesgos*: utiliza estrategias para compartir recursos en la *CS* de manera que el riesgo de una interrupción del suministro sea minimizado.
- *CS ágiles*: utilizan estrategias dirigidas a ser sensibles y flexible a las necesidades de los clientes, mientras que el riesgo a las roturas de stocks o interrupciones es evitado por medio de Inventarios y otros recursos de Capacidad. Estas CS combinan las fortalezas de las CS de Respuesta Rápida y con Protección contra Riesgos.

Childerhouse et al. (2002) estudian los productos de una industria de iluminación, sus Ganadores y Calificadores de Pedidos (*Order Winners* y *Order Qualifiers*) y las etapas del ciclo de vida en las que se encuentran los productos. Proponen las estrategias de CS más apropiadas para los distintos productos (ver la figura 2.4).

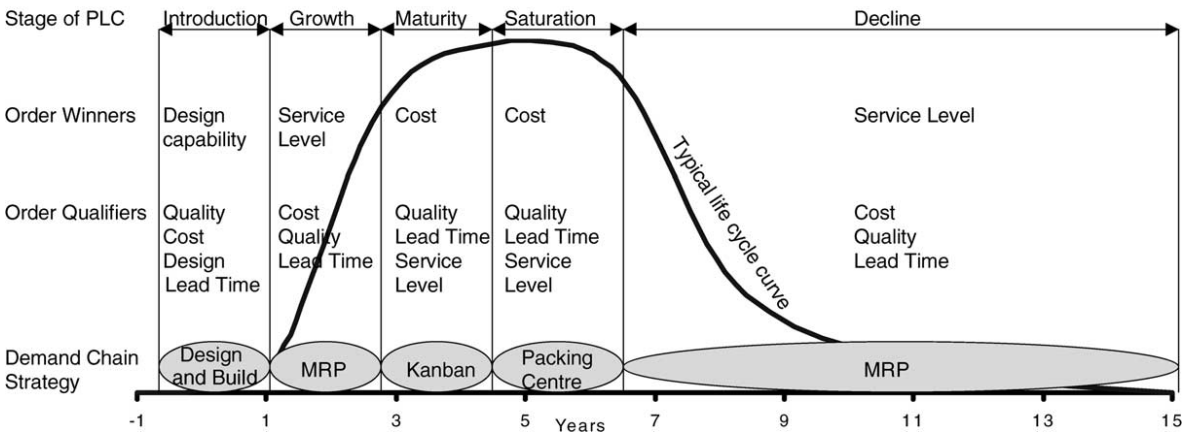


Figura 2.4: Ciclo de vida y estrategias de CS en una industria de iluminación (Childerhouse et al., 2002)

- Durante la etapa de Introducción del Ciclo de Vida del Producto, la Capacidad de Diseño es la principal *Order Winner* (*OW*), por lo tanto la estrategia de CS de Diseño y Construcción (*Design and Built*) es la más apropiada (ver la figura 2.5).

- En la etapa de Crecimiento, el Nivel de Servicio en términos de disponibilidad del producto con demanda imprevisible es la principal *OW*, por consiguiente la estrategia MRP[2] de CS es la más apropiada Para los productos que están en madurez debe emplearse la estrategia Kanban de CS, para competir mejor en base al coste como *OW* clave.

- Durante la etapa de saturación, la empresa compite ofreciendo múltiples variantes de productos; para lograrlo, la estrategia de Centro de Empacado *(Packing Centre)* es la más adecuada.

- Finalmente, en la etapa de declive, los productos deben ser producidos nuevamente con la estrategia MRP para maximizar el nivel de servicio correspondiente al volumen bajo de cada modelo y demanda sumamente imprevisible.

[2] Los nombres elegidos por los autores se han mantenido aunque en ocasiones pueden suponer más un conflicto que una ayuda.

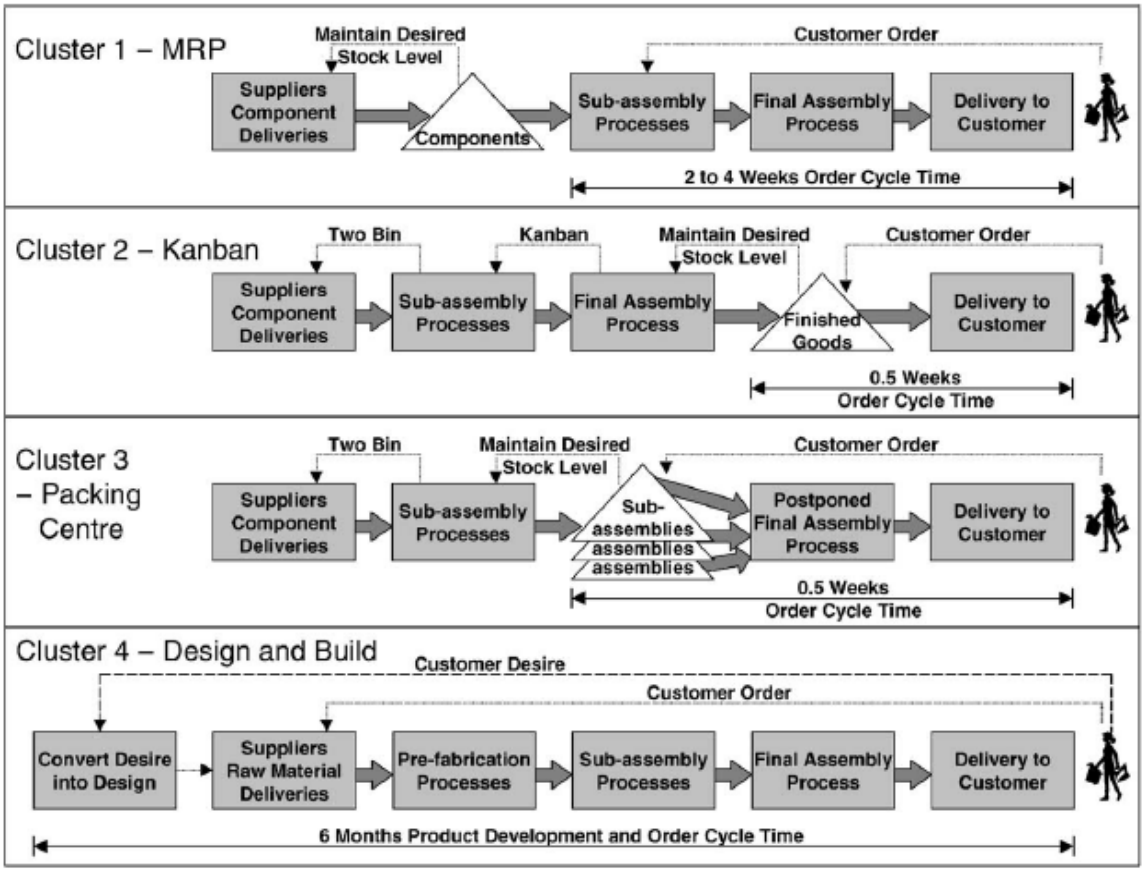


Figura 2.5: Cuatro estrategias de CS (Childerhouse et al., 2002)

Stratton y Warburton (2006) dicen que el principal problema del Aprovisionamiento de un país lejano es el Plazo de Entrega. Muestran los tiempos de entrega (*supply lead-time*) típicos para abastecimiento global (ver la tabla 2.4).

Rubro	Tiempo (semanas)
Ropa por barco	6
Manufactura	6
Hilo por barco	4
Hilo y color (trabajo)	4-6

Tabla 2.4: Tiempo total de entrega para suministro global por rubro
(Stratton y Warburton, 2006)

Los autores mencionan que, en el caso de Griffin Manufacturing, los proveedores de Honduras requerían que la Previsión de la Demanda se hiciera con 16 semanas de anticipación a la estación de ventas. En consecuencia los errores de

Previsión comúnmente excedían el 25%. La precisión de la Demanda se podía mejorar significativamente sólo cuando empezaba la estación (estación de ventas de 12 a 16 semanas). Esto obligó a Griffin a separar en tiempo sus órdenes y tener dos momentos: 16 semanas de anticipación para la producción de Honduras y 6 semanas para la producción local. En la tabla 2.5 se resume la estrategia empleada por Griffin.

Medios para reducir la incertidumbre	**Estrategias de separación**	**Uso estratégico de la Capacidad e Inventarios**
Usar los datos de Ventas del inicio de la temporada para mejorar la precisión de la Previsión antes de lanzar la Orden a la producción local.	Separar los requerimientos en conflicto en Tiempo (órdenes tempranas y tardías) y en Espacio (producir en Honduras por *bajo coste* y en Griffin para *respuesta rápida*).	Subcontratar 85% de las prendas al Proveedor de bajo coste de Honduras contra *Previsión adelantada de la Demanda.* Producir en Griffin el 15% mediante *Reserva de Capacidad* e Inventario de Materiales.

Tabla 2.5: Estrategia de Griffin Manufacturing (Stratton y Warburton, 2006)

Christopher et al. (2006) toman en cuenta que la precisión de la Previsión de la demanda es menor a medida que la anticipación es mayor. En la industria de la moda, las decisiones de estilo, color y cantidad tienen que ser tomadas varios meses antes de la estación y por ello la posibilidad de error en el pronóstico es grande. Los errores de Previsión en la industria de la moda en función del horizonte de tiempo, a "ojo de buen cubero" (*rule of thumb* en el original) según Blackburn (1991), son:

- Al inicio de la estación ± 10 por ciento;
- 16 semanas antes ± 20 por ciento; y
- 26 semanas antes ± 40 por ciento.

Además, los autores consideran importante el Tiempo de Entrega; Lowson (2001) ha sugerido que para la industria detallista de ropa del Reino Unido, el cambio a fuentes de suministro de ultramar (*offshore* en el original) puede cuadriplicar el tiempo requerido para una orden de envío. En función de la variación de la demanda (estable o volátil), el tiempo de reaprovisionamiento (*replenishment lead-time* en el original) corto o largo, y el tipo de producto (estándar o especial), los autores dicen que podría haber ocho (2 x 2 x 2) CS teóricas, pero en la práctica algunas son improbables o no viables. En la figura 2.6 presentan una matriz con las cuatro CS sugeridas como resultado de su clasificación.

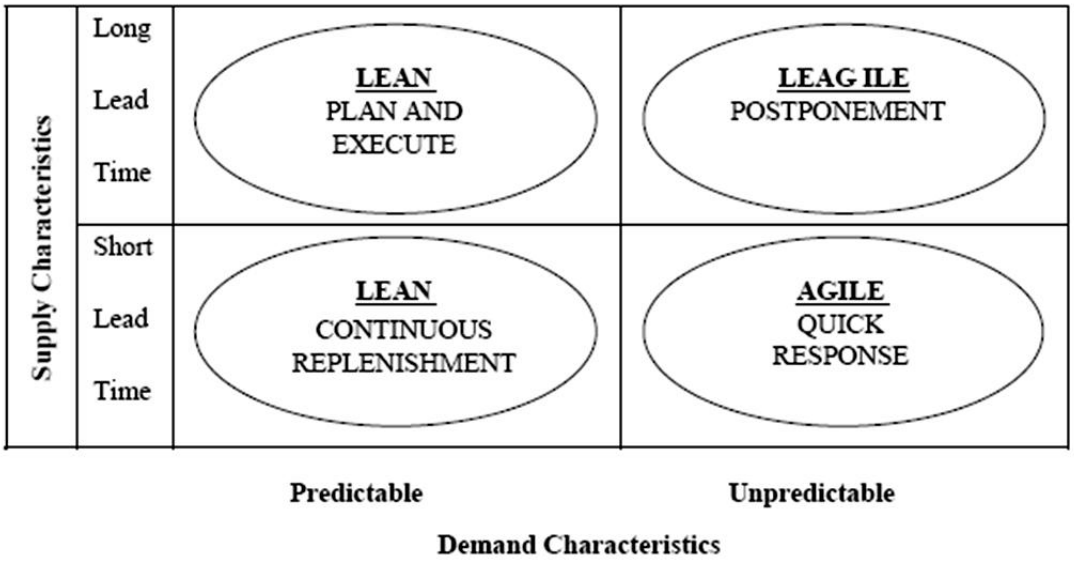


Figura 2.6: Clasificación de las cadenas de suministro y su estrategia
(Christopher et al., 2006)

En la esquina superior derecha se ubica la CS *leagile* para demanda impredecible y largos tiempos de reaprovisionamiento; esta CS debe mantener Inventarios en alguna forma genérica y ensamblar/configurar/distribuir a medida que la demanda real lo requiera. Esto es el clásico concepto de aplazamiento (*postponement*). Hewlett Packard siguió esta estrategia para su línea de impresoras DeskJet.

Cuando la demanda es imprevisible y los plazos de entrega son cortos (esquina inferior derecha de la figura), se requerirá una CS ágil capaz de dar una respuesta rápida (*Quick Response*).

En la esquina superior izquierda, para demanda predecible y plazo de entrega largo, se recomienda la CS lean con *Plan and Execute* (P&E). P&E es el

sistema *push* por el que se lanzan las Órdenes de Compra y Producción con mucha anticipación en función de Previsiones de la Demanda.

Por último, en la esquina inferior izquierda, para demanda predecible y plazo de entrega corto, se recomienda la CS *lean* pero con *Continuous Replenishment* (CR). CR es apropiado para productos maduros con demanda uniforme de manera que se asegura un abastecimiento continuo a los detallistas.

Wong et al. (2006) han examinado la CS de una compañía de juguetes y han encontrado que los juguetes son mayormente productos innovadores o intermedios pero no funcionales. Los autores afirman que los productos con demanda volátil, como ropa de moda y juguetes, con ciclos de vida relativamente cortos (de 6 meses a 2 años) y sujetos a alta estacionalidad e incertidumbre de la demanda, no han sido estudiados suficientemente. Proponen una nueva forma de diferenciar los productos usando cuatro determinantes de riesgo: incertidumbre en las Previsiones, variabilidad de la Demanda, Margen de contribución y Tiempo de Atención (*time window of delivery* en el original), y con estos criterios distinguen los productos para diseñarles una CS adecuada. Agregan un nuevo tipo de CS al modelo de Fisher (1997), la CS de Respuesta Rápida para Productos Intermedios (*Physically Responsive for Intermediate Products*).

Los autores indican que el objetivo de la *CS physically responsive* es de suministrar productos con el Inventario adecuado para brindar altos Niveles de Servicio y cumplir los Plazos de Entrega. La *CS physically efficient* debe satisfacer la demanda predecible al menor coste posible. La *CS market responsive* debe responder rápidamente a la Demanda impredecible minimizando roturas de stock, rebajas de fin de temporada e inventario obsoleto. La figura 2.7 muestra las estrategias propuestas (ver en la siguiente página).

En la figura 2.7 se puede apreciar que la *CS physically responsive* (de productos intermedios) tiene su punto principal de stock en el Centro de Distribución (DC en la figura), desde donde atiende la demanda de los detallistas (*Retailers*). En cambio, la *CS market responsive* (de productos innovadores), hace un envío inicial de productos a los detallistas (antes del inicio de la temporada de ventas) a partir de su stock en el Centro de Distribución (este stock debe haberse producido con la estrategia de producción MTS), y luego, en plena temporada de ventas, hace envíos subsiguientes

de productos que se ensamblan a partir de un stock de componentes (siguiendo la estrategia de producción ATO). En la figura, además se ve que para pedidos especiales y productos para eventos se propone emplear la estrategia de producción MTO (que debería llamarse ETO porque incluye el diseño).

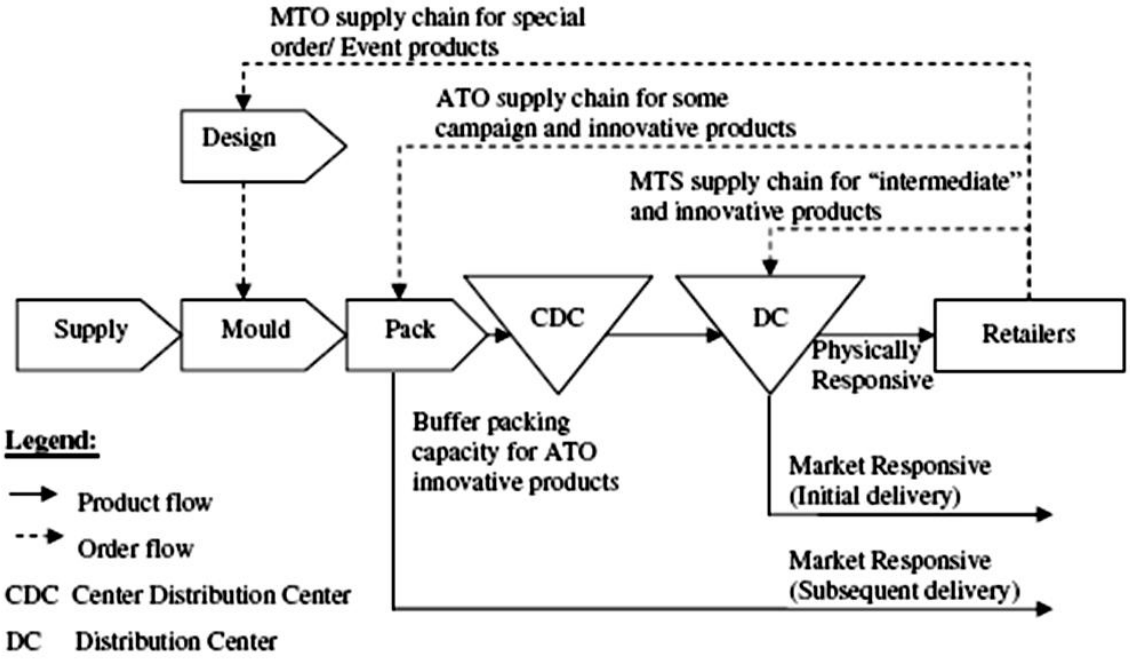


Figura 2.7: Estrategias propuestas (Wong et al., 2006)

Vonderembse et al. (2006) han realizado un estudio sobre tres tipos de productos: estándar, innovadores e híbridos. Los productos estándar e innovadores coinciden con los identificados por Fisher (1997). Los productos híbridos son productos complejos integrados por muchos componentes los cuales pueden ser a su vez productos innovadores y estándar. El automóvil es un ejemplo de este tipo de productos. Analizan la relación de los productos identificados con los tipos de CS (ajustada, ágil e híbrida) en función de la fase del Ciclo de Vida en que se encuentran. Concluyen afirmando que los productos estándar deben producirse siempre en una CS ajustada independientemente de la etapa del ciclo de vida en que se encuentren; que los productos híbridos deben fabricarse siempre en CS híbridas, y los productos innovadores en CS ágiles para las etapas de introducción y crecimiento, y en CS ajustadas para las etapas de madurez y declive (ver la tabla 2.6).

<table>
<tr><th>Tipo de Producto
Ciclo de Vida</th><th>Estándar</th><th>Innovador</th><th>Híbrido</th></tr>
<tr><td>Introducción</td><td rowspan="4">CS ajustada</td><td rowspan="2">CS ágil</td><td rowspan="4">CS híbrida</td></tr>
<tr><td>Crecimiento</td></tr>
<tr><td>Madurez</td><td rowspan="2">CS ajustada</td></tr>
<tr><td>Declive</td></tr>
</table>

Tabla 2.6: Clasificación de las CS en función del tipo de producto y del ciclo de vida (Vonderembse et al., 2005)

Kumar y Arbi (2008) analizan mediante simulación de Monte Carlo una cadena de suministro con subcontratación de la producción (del producto completo) a fábricas en países lejanos de menor coste de manufactura. Emplean datos de empresas reales y consideran incertidumbre en el tiempo de proceso de la orden, la producción, el transporte, el coste, la calidad, etc. Concluyen que gracias a la subcontratación se puede ahorrar el 26% de los costes (con un mínimo de 18% en el peor de los escenarios) pero estadísticamente el tiempo promedio de aprovisionamiento (lead time) fluctúa entre 41 y 91 días con una media de 57 días (esto porque se consideró un tiempo de transporte en barco de 36 a 48 días). Como resultado, afirman los autores, la subcontratación (en un país lejano) no es una solución viable para productos de demanda de corto plazo; sin embargo, para órdenes de estaciones largas la subcontratación puede ser un gran ahorrador de costes.

Romano (2009) compara las CS de Zara y Benetton en función de su desempeño en Tiempo (su plazo de entrega total). Identifica que Zara emplea dos CS: una Ajustada con proveedores lejanos (de Asia y México) para sus productos funcionales (*basic collection*) y otra Ágil con proveedores locales (de Europa y norte de África) para sus productos innovadores (*flash collection*). Así mismo, identifica que Benetton emplea tres CS: la primera Ajustada con proveedores lejanos (de Asia y el Lejano Este) para lograr bajo coste, la segunda Ajustada pero con proveedores menos lejanos (en Croacia) para lograr alta calidad, y la tercera Ágil con proveedores locales (de Europa) para lograr respuesta rápida a las tendencias del mercado.

A continuación se muestra una tabla comparativa (tabla 2.7) de las Estrategias puras de CS, indicando las equivalencias entre las distintas denominaciones empleadas por los diferentes investigadores (en la tabla 2.7, las estrategias de cada renglón son equivalentes). Se denomina Estrategias puras a aquellas que consideran sólo una CS pero con el punto de desacople ubicado en un eslabón distinto de la cadena.

Hoekstra y Romme - 1992	Fisher - 1997	Naylor et al. – 1999 Mason-Jones et al. – 2000	Childerhouse et al. - 2002	Wong et al. – 2006	Vonderembse et al. - 2006
BTO	---	---	*Design and Build*	---	---
MTO	---	Ágil	*MRP*	---	Ágil
ATO	Respuesta Rápida	*Agiltada*	*Packing Centre*	*Physically responsive*	Híbrida
STS	Eficiente	Ajustada	*Kanban*	*Physically efficient*	Ajustada

Tabla 2.7: Clasificación de las *CS puras* por autores (elaboración propia)

En la tabla 2.7 no aparece directamente la aportación de Lee (2002) pero sí aparece indirectamente porque su CS de Protección contra Riesgos es igual a la CS Eficiente de Fisher. La CS de Protección contra Riesgos sólo se diferencia de la CS Eficiente de Fisher en que emplea varios Proveedores sustitutos (dividiendo sus Pedidos entre varios Proveedores del mismo material o componente) de manera que se minimiza el riesgo de desabastecimiento.

En la tabla 2.8 se muestran las estrategias híbridas propuestas por distintos autores. Se denomina estrategias híbridas a aquellas que consideran dos o más CS. En este caso no hay equivalencia entre sus propuestas (ver la tabla en la siguiente página).

Hago notar que la CS de Respuesta Rápida de Fisher (1997) no coincide con la estrategia híbrida Base e Imprevista (B&I) de Christopher y Towill (2001) porque la CS de Respuesta Rápida es una sola CS tipo *agiltada* (con el punto de desacople antes del ensamble final), mientras la estrategia B&I implica el uso de dos CS, una ajustada y otra ágil. La CS de Respuesta Rápida requiere productos modulares (es decir que se

puedan fabricar los componentes y almacenar hasta su posterior ensamblado) mientras la estrategia B&I no necesita que los productos tengan estructura modular.

Christopher y Towill - 2001	Wong et al. – 2006	Romano – 2009
Pareto 80/20	---	**Zara**: *Lean* (Prov. lejano) y *Agile* (Prov. local y cercano)
Base e Imprevista	Market Responsive	**Benetton**: *Lean* (Prov. muy lejano), *Lean* (Prov. poco lejano) y *Agile* (prov. local)

Tabla 2.8: Clasificación de las CS híbridas por autores (elaboración propia)

En la tabla 2.9 (ver en la siguiente página) se presenta un resumen de las Estrategias de CS y de Producción en función del tipo de Producto, tipo de Demanda, Ganador de Mercado (GM) y Proveedores existentes.

Producto/ Demanda	**Ganador de mercado (GM) y Estrategia de la CS (ECS)**	**Proveedor**	**Estrategia de Producción**
Producto funcional Demanda predecible	GM: ***Coste***. ECS: ***Eficiente, Ajustada***.	Local: si los costes de los Proveedores locales son iguales o menores que los de los Proveedores extranjeros.	***MTS-Kanban***: asegura plazo de entrega corto.
		Extranjero: si los costes totales de los Proveedores extranjeros son menores que los de los locales.	***MTS-P&E***: para plazo de entrega largo.
Producto innovador Demanda impredecible	GM: ***Disponibilidad de productos*** en el momento oportuno. ECS: de ***Respuesta Rápida***, ***Agiltada***, ***Base e Imprevista***, ***Market Responsive***.	Local: sólo si sus Costes son iguales o menores que los de los Proveedores extranjeros.	***MTO***: cuando el proceso permite un plazo de entrega corto.
		Local y Extranjero: si los Costes del extranjero son menores que los del local, pero el Plazo de Entrega del extranjero es mayor que el del local.	***MTS-MTO***: cuando el Producto *no* tiene estructura *modular.* ***MTS-ATO***: cuando el *Producto es modular*.
Fabricado contra pedido Demanda impredecible	GM: ***Flexibilidad y Confiabilidad***. ECS: ***Ágil.***	Proveedor local o extranjero, dependiendo del plazo de entrega, el coste y la calidad.	***MTO***: fabricar componentes y ensamblar *contra Pedido*.
Diseñado contra pedido Demanda impredecible	GM: ***Capacidad de Diseño.*** ECS: con ***Punto de Desacople en el Diseño.***	Proveedor local, y sólo si no hay Proveedor local entonces Proveedor extranjero.	***ETO:*** diseñar **contra Pedido** y puede incluir la fabricación *contra Pedido-(**MTO**)*.

Tabla 2.9: Resumen de las Estrategias de CS y de Producción para distintos Tipos de Producto / Demanda, Ganador de Mercado y Proveedor (elaboración propia)

Dada la clasificación de Lamming et al. (2000) de productos de alta y baja complejidad y su correspondiente Ganador de Mercado (GM), y considerando lo indicado por Naylor et al. (1999) y Mason-Jones et al. (2000), podemos deducir el tipo de CS que le corresponde a cada tipo de producto:

	Alta complejidad	Baja complejidad
Productos innovadores	GM: flexibilidad CS: ágil	GM: tiempo de entrega CS: agiltada
Productos funcionales	GM: coste CS: ajustada	GM: coste CS: ajustada

Tabla 2.10: Estrategias de CS según tipo de Producto, complejidad y Ganador de Mercado (elaboración propia)

Se puede ahora responder a las preguntas iniciales de este apartado:

- ¿Existen actualmente diferentes estrategias de las CS?
- ¿En qué circunstancias se debe emplear cada una?

En la tabla 2.11 se responde a las preguntas agregando el tipo de Estrategia de Producción (ver la tabla en la siguiente página).

Estrategias de las CS	**Estrategia de Producción**	**Circunstancias de empleo**
Puras:		
Ajustada	MTS: Kanban y P&E	Producto funcional y proceso estable.
Con protección contra riesgos	MTS	Producto funcional y alta incertidumbre en el Aprovisionamiento.
Agiltada	ATO	Producto innovador con diseño modular y plazo de entrega largo.
Ágil	MTO	Producto innovador con plazo de entrega corto.
Híbridas:		
Pareto 80/20	MTS y MTO	Productos funcionales e innovadores. No requiere diseño modular. Proveedores lejanos y locales.
Base e Imprevista	MTS y MTO	Productos innovadores. No requiere diseño modular. Proveedores lejanos y locales.
Market Responsive	MTS y ATO	Productos innovadores. Diseño modular. Proveedores lejanos y locales.

Tabla 2.11: Estrategias de la CS y de Producción y cuándo deben emplearse (elaboración propia)

Para responder a la pregunta:

¿Cómo se realiza actualmente la planificación táctica de las operaciones de CS de respuesta rápida sin/con estructura alternativa de procesos?

Debemos recordar primeramente que la CS de respuesta rápida es para productos innovadores, los cuales tienen ciclo de vida corto y alta incertidumbre en la demanda.

Además debemos considerar que actualmente existen proveedores locales (con plazo de entrega corto y coste de producción alto) y proveedores lejanos (con plazo de entrega largo y coste de producción bajo), ambos con procesos estables.

Entonces, la CS puede emplear la estrategia Base e Imprevista junto con la *Market Responsive* en la medida que los productos tengan diseño modular.

La explicación es la siguiente: se hace la previsión de la demanda con mucha anticipación y se separa la parte Base de la parte Imprevista (*surge*) de la demanda. Luego se planifica la producción de la demanda Base con anticipación y se subcontrata al proveedor lejano de bajo coste. En cuanto a la demanda *surge*, ésta se separa en componentes y producto final. Los componentes a su vez se separan en comunes y no comunes. Los comunes se subcontratan al proveedor lejano con anticipación y son remitidos al proveedor local, dejando el ensamble final para realizarse localmente en plena temporada de ventas. Los componentes no comunes los fabrica el proveedor local en función de una previsión más precisa de la demanda al inicio de la temporada de ventas. Si los productos no tienen diseño modular, el proveedor local satisface toda la demanda *surge* en función de una previsión más precisa de la demanda al inicio de la temporada de ventas y en función de pedidos en firme durante la misma temporada, con la estrategia ágil.

De esta manera, actualmente se lanza al mercado un primer lote de productos (producidos con anticipación, contra previsiones, por proveedores lejanos) y luego se realizan envíos subsiguientes de productos ensamblados por el proveedor local (a partir del stock local de componentes). Cuando los productos no tienen estructura modular, el proveedor local produce completamente los productos de los envíos subsiguientes.

En el siguiente apartado se ven los requisitos para la planificación táctica de la producción en cadenas de suministro de productos innovadores considerando las características indicadas arriba.

2.3. Requisitos para la planificación de la producción en cadenas de suministro de productos innovadores

La Planificación de la CS se da en tres niveles: estratégico, táctico y operativo (Shapiro (1998), Chopra y Meindl (2001), Van Landeghem y Vanmaele (2002)). Fleischmann et al. (2005) y Stadtler (2005) clasifican las tareas de Planificación en planificación de: largo plazo, medio plazo y corto plazo. La matriz de planificación de la CS clasifica las tareas de planificación en dos dimensiones "horizonte de planificación" y "proceso de la CS". La tabla 2.12 muestra las tareas típicas que ocurren en la mayoría de las cadenas de suministro. Las tareas de largo plazo están incluidas en un solo rectángulo para mostrar el carácter integrador de la planificación Estratégica.

Horizonte de Planificación	**Procesos de la CS**			
	Aprovisionamiento	**Producción**	**Distribución**	**Ventas**
Largo Plazo	• Programa de Materiales • Elección de Proveedores • Cooperación	• Localización de plantas • Sistema de producción	• Estructura física de la distribución	• Programa de producto • Planificación estratégica de ventas
Medio Plazo	• Planificación de personal • MRP • Contratos	• Plan Maestro de Producción • Planificación de la Capacidad	• Planificación de la Distribución	• Planificación de Ventas de mediano plazo
Corto Plazo	• Programación de Personal • Pedidos de Materiales	• Tamaño de Lotes • Programación de Máquinas • Control en planta	• Reposición de Inventarios • Programación del Transporte	• Programación de Ventas de corto plazo

Tabla 2.12: La matriz de planificación de la CS (Fleischmann et al., 2005)

De las tareas de Planificación descritas en la matriz de planificación de la CS, las que corresponden a la Planificación de la Producción en el medio plazo son el Plan Maestro de Producción (PMP) y la Planificación de la Capacidad Aproximada –PCA- (*Rough-Cut Capacity Planning*). Según Fleischmann et al. (2005) estas tareas deben planificar de manera eficiente el uso de la Capacidad de producción disponible en una o más instalaciones. El plan resultante se basa en Familias de productos y en periodos semanales o mensuales, sin considerar cada proceso de producción individual. El objetivo es balancear el coste de la capacidad (horas regulares, horas extra, tiempo ocioso, contratación y despido, subcontratación) contra el coste de los inventarios ante una demanda con fluctuaciones estacionales. Además, si se considera más de una instalación de producción, los costes de transporte entre las localidades tienen que ser incluidos en la función objetivo.

Actualmente las CS abarcan empresas en distintos países, que planifican y coordinan sus capacidades (de producción, almacenamiento, transporte, etc.) para brindar al mercado productos en el tiempo, cantidad, variedad, calidad y coste adecuados, por ejemplo la industria de la confección tiene el diseño, producción de telas, corte, costura y venta al por menor en diferentes partes del mundo (Fisher et al. 1994). Muchas otras industrias, tales como equipo de telecomunicaciones y ordenadores, electrónica de consumo y juguetes, tienen una Estructura de Cadena de Suministro semejante debido a los bajos costes de mano de obra de países del sudeste asiático, China, Centro América, etc. (Christopher et al., 2006). Esto obliga a planificar la producción con visión de cadena de suministro (abarcando dos o más niveles -montaje final, subensambles, fabricación de componentes, etc.- en dos o más empresas ubicadas en distintos países) y a considerar la estrategia de CS más adecuada según las características de los productos, la demanda y los proveedores (plazos de entrega, costes, etc.).

Los Proveedores de bajo coste exigen que se comprometa la capacidad producción con mucha anticipación a la temporada de ventas (Stratton y Warburton, 2003 y 2006), y permiten luego el uso de horas extra con una penalización proporcional. Sin embargo, si no se usa la capacidad contratada no hay descuento alguno.

Entonces, para efectos de planificación de la producción, se tiene que hacer un primer Plan Maestro de Producción (PMP) con mucha anticipación a la temporada

de ventas y contratar la capacidad de producción de los proveedores lejanos durante tres a seis meses, y luego, al inicio de la temporada de ventas, hacer un segundo PMP para redefinir la producción de los meses siguientes (hasta cuando acaba la temporada de ventas) tanto en los Proveedores lejanos como en los Proveedores locales.

Por todo lo anterior y lo visto en el Apartado 2.2, un sistema de planificación para calcular el mejor PMP posible para una CS de productos innovadores debe tener en cuenta las siguientes características:

- Abarcar al menos un ciclo completo de demanda (el cual depende de la duración de la temporada de ventas) y planificar en periodos semanales o quincenales (dependiendo de la mínima unidad de tiempo requerida para el flujo de productos).

- Abarcar distintos nodos de proveedores lejanos y de proveedores locales, ubicados en distintos países o regiones, considerando sus diferentes capacidades, plazos de entrega y costes.

- Planificar de manera eficiente el uso de la capacidad de producción en dos o más instalaciones; incluyendo horas extras y tiempo ocioso.

- Planificar las cantidades a producir de cada uno de los productos finales, subensambles, componentes y partes, en cada nodo y en cada proceso; esto implica tener dos o más niveles en la Lista de Materiales de cada producto final. Además debe poder tener componentes comunes entre distintos productos finales (modularidad), lo mismo que materias primas comunes. Y debe poder planificar las cantidades a comprar de cada materia prima en cada nodo.

- Considerar los tiempos de producción y los tiempos de transporte desde los proveedores lejanos hasta el país del contratista considerando varias alternativas de transporte (avión, barco, etc.).

- Considerar la incertidumbre de la demanda y la anticipación con la cual se deben colocar los pedidos en firme a los proveedores lejanos y locales; y debe poder replanificar la producción (empleando previsiones de demanda más precisas).

- Considerar la posibilidad de aplazamiento (*postponement*), de manera que se puedan fabricar los productos de dos maneras: una directa, desde la materia prima hasta el producto final (sin inventarios intermedios), y de manera indirecta, es decir produciendo independientemente lotes de los distintos componentes y subensambles para almacenarlos y luego, en otro periodo e incluso puede ser en otro nodo, hacer el ensamble final (estrategia *leagile*).

- Considerar los *costes* de: inventario, entregas diferidas (también llamadas "pedidos pendientes"), producción, materias primas, transporte, capacidad contratada, horas extra y tiempo ocioso.

- Considerar los ingresos por ventas y el valor residual de los productos al final de la temporada (ingresos por remate de productos terminados).

En el siguiente apartado se ven los aportes de distintos investigadores relacionados con la Planificación Táctica de la Producción en cadenas de suministro de productos tanto Funcionales como Innovadores.

2.4. Modelos de Programación Matemática de la Planificación de la Producción en Cadenas de Suministro de Productos Innovadores

En el apartado anterior se han establecido los requerimientos de un sistema que permita la planificación de un modo automático. Un modo habitual de diseñar dicho sistema es a partir de modelos de programación matemática. En este apartado se revisan algunos de los más relevantes con dicho objetivo.

Se ha seleccionado sólo los artículos que presentan modelos de Planificación de la producción a nivel Táctico, Estratégico-Táctico o Táctico-Operativo, y se han clasificado en dos grupos, aquellos con todos sus datos y parámetros deterministas (tabla 2.13a) y aquellos con incertidumbre de demanda, costes y/o tiempos (tabla 2.14a). Se ha determinado para cada artículo si su modelo abarca el Plan Maestro de Producción (PMP), la Planificación de la Capacidad Aproximada (PCA) y/o la Planificación de la Distribución (PD), y el tipo de modelo matemático empleado (PL, PEM, PLEM, PNL, *fuzzy* y/o simulación). En las tablas 2.13b y 2.14b se muestran los

nombres de los autores y de los artículos en el orden en que son presentados en las tablas 2.13a y 2.14a.

De las muchas características que se podrían identificar en los modelos, se ha elegido aquellas que se consideran necesarias para la planificación táctica de las operaciones en CS de productos innovadores, como son: si el modelo considera dos o más momentos de decisión para comprometer la producción en firme, las etapas de la CS comprendidas (Proveedores, Fabricantes, Distribuidores, Detallistas, Clientes –P-F-D-Det-C-), el número de niveles abarcados (se ha asignado N cuando el número de niveles comprendido puede ser mayor que 3), los costes (escasez, inventario, set-up, producción, materias primas, transportes y entregas diferidas), las capacidades (de las plantas, almacenes, proveedores, máquinas, horas extra, subcontratación y tiempo ocioso), diferentes modos de transporte (barco, avión, etc.) y si el modelo permite la producción directa e indirecta (con aplazamiento).

En las tablas se ha puesto "X" cuando la característica es considerada en el modelo matemático y "-" cuando la característica no es considerada.

El número de "Aspectos abarcados" (ver las tablas 2.13a y 2.14a) considera desde Varios Momentos de Decisión hasta Producción directa e indirecta, por lo tanto son 19. Sólo un artículo abarca 15 de estos aspectos, el resto abarca 12 o menos aspectos.

En la tabla 2.13a se muestran los artículos cuyos modelos son deterministas. Se puede apreciar que sólo uno emplea horizonte rodante en su programa matemático, sólo cuatro abarcan dos o más etapas de la CS y sólo cuatro comprenden dos o más niveles (el artículo 9 tiene 2 niveles pero el material del nivel inferior tiene una relación de 1 a 1 con el de nivel superior y no emplea la lista de materiales). Sólo un artículo abarca varias etapas y varios niveles a la vez. Sólo dos artículos consideran varios medios de transporte y sólo un artículo abarca la posibilidad de fabricar tanto de forma directa como indirecta. Dos aspectos no incluidos en la tabla son los tiempos de transporte y el coste de contratación y despido; seis artículos consideran los tiempos de transporte y cuatro incluyen los costes de contratación y despido. Los artículos que más aspectos abarcan corresponden a Arntzen et al. (1995) y a Chern y Hsieh (2007).

	1	2	3	4	5	6	7	8	9	10	11	12	Total
Tareas de planificación	PMP	PMP-PCA	PMP-PCA-PD	PMP-PCA-PD	PMP	PMP	PMP	PMP-PD	PMP-PCA	PMP	PMP	PMP	12
Modelo	PLEM	PEM	PEM	PL y Sim	PNL	PL Fuzzy	PLEM	PL	PLEM	PD	PLEM	PL Fuzzy	12
Varios Momts de Dec	-	-	-	-	-	-	-	X	-	-	-	-	1
Etapas	P-F-D-C	Fab	Fab	Fab	Fab	Fab	Fab	Fab	Fab	P-F	P-F-D	F-D	4
Niveles	N	1	1	3	1	1	1	3	2	1	1	1	4
Costes													
Escasez	-	X	-	X	X	-	-	-	-	-	-	-	3
Inventario almacenes	X	X	X	X	X	X	X	X	X	X	X	-	11
Set-up	X	X	-	-	-	-	-	-	X	-	-	-	3
Producción (variable)	X	X	X	X	X	X	X	X	X	X	-	X	11
MP (coste de compra)	-	-	X	X	X	-	-	X	-	-	-	-	4
Transporte	X	-	X	X	X	-	-	X	-	X	X	X	8
Entregas diferidas	-	X	-	X	-	X	X	X	X	X	-	-	7
Capacidades													
Plantas (máxima)	X	X	X	X	X	X	X	X	X	-	X	X	11
Almacenes	X	-	X	X	-	X	X	X	X	-	X	X	9
Proveedores	X	-	X	-	-	-	-	X	-	X	-	-	4
Máquinas o líneas	X	X	X	X	-	X	-	-	X	-	-	X	7
Horas extra	-	X	-	-	-	X	X	-	X	-	-	-	4
Subcontratación	-	X	X	-	-	X	X	X	-	X	-	-	6
Tiempo ocioso	-	X	-	-	-	-	-	-	-	-	-	-	1
Otros													
Modos transporte	X	-	X	-	-	-	-	-	-	-	-	-	2
Prod directa e indirecta	-	-	-	-	-	-	-	-	X	-	-	-	1
Aspectos abarcados	11	10	9	10	6	8	7	11	9	7	5	6	

Tabla 2.13a: Artículos con modelos deterministas

1	Arntzen, B.C., Brown, G.G., Harrison, T.P. y Trafton, L.L. (1995). Global Supply Chain Management at Digital Equipment Corporation. *Interfaces* 25(1), 69-93.
2	Baykasoglu, A. (2001). MOAPPS 1.0: aggregate production planning using the multiple-objective tabu search. *International Journal of Production Research,* 39(16), 3685-3702.
3	Goetschalckx, M., Vidal, C.J. y Dogan, K. (2002). Modeling and design of global logistics systems: A review of integrated strategic and tactical models and design algorithms. *European Journal of Operation Research*, 143(1), 1-18.
4	Lee, Y.H., Kim, S.H. y Moon, Ch. (2002). Production-distribution planning in supply chain using a hybrid approach. *Production Planning & Control*, 13(1), 35–46.
5	Jackson, J.R., y Grossmann, I.E. (2003). Temporal decomposition scheme for nonlinear multisite production planning and distribution models. *Industrial and Engineering Chemistry Research*, 42(13), 3045–3055.
6	Wang, R.C. y Liang, T.F. (2004). Application of fuzzy multiobjective linear programming to aggregate production planning. *Computers and Industrial Engineering*, 46 (1), 17–41.
7	Gomes da Silva, C., Figueira, J., Lisboa, J. y Barman, S. (2006). An interactive decision support system for an aggregate production planning model based on multiple criteria mixed integer linear programming. *Omega*, 34 (2), 167 – 177.
8	Chern, C.-C. y Hsieh J.-S. (2007). A heuristic algorithm for master planning that satisfies multiple objectives. *Computers & Operations Research*, 34(11), 3491-3513.
9	Leung, S.C.H. y Ng, W-L (2007a). A goal programming model for production planning of perishable products with postponement. *Computers & Industrial Engineering*, 53 (3), 531-541.
10	Sargut, F. Z. y Romeijn, H. E. (2007). Capacitated production and subcontracting in a serial supply chain. *IIE Transactions* 39, 1031–1043.
11	Kanyalkar, A.P. y Adil, G.K. (2007). Aggregate and detailed production planning integrating procurement and distribution plans in a multi-site environment. *International Journal of Production Research*, 45(22), 5329-5353.
12	Liang, T.-F. (2008). Integrating production-transportation planning decision with fuzzy multiple goals in supply chains. *International Journal of Production Research*, 46 (6), 1477–1494.

Tabla 2.13b: Autores y artículos de la Tabla 2.13a

Entre los modelos deterministas no hay ninguno que considere a la vez: varios momentos de decisión, varias etapas de la CS, varios niveles de materiales, varios modos de transporte y sus tiempos, y la posibilidad de fabricar tanto de forma directa como indirecta.

En la tabla 2.14a se muestran los artículos cuyos modelos consideran la incertidumbre de la demanda y/o los tiempos y/o los costes. Se puede apreciar que cuatro emplean varios momentos de decisión en su programa matemático, ocho abarcan dos o más etapas de la CS y seis comprenden dos o más niveles (pero el artículo 2 considera que todos los productos terminados tienen un mismo componente, y en el artículo 8 el material del nivel inferior tiene una relación de 1 a 1 con el producto de nivel superior; estos dos artículos no emplean la lista de materiales). Sólo dos artículos consideran distintos medios de transporte y solamente uno abarca la posibilidad de fabricar tanto de forma directa como indirecta. Dos aspectos no incluidos en la tabla son los tiempos de transporte y el coste de contratación y despido; sólo dos artículos consideran los tiempos de transporte (que son justamente los dos que abarcan distintos medios de transporte) y tres incluyen los costes de contratación y despido. El artículo que abarca más aspectos es el de Peidro et al. (2007) pero no abarca la fabricación directa e indirecta.

Entre los modelos con incertidumbre no hay ninguno que considere a la vez: varios momentos de decisión, varias etapas de la CS, varios niveles de materiales, varios modos de transporte y sus tiempos, y la posibilidad de fabricar tanto de forma directa como indirecta; todos estos son aspectos necesarios para la planificación táctica de la producción de productos innovadores.

De todos los artículos seleccionados, sólo Barbarosoglu (2000) considera que los pedidos del fabricante al proveedor deben ser hechos en distintos periodos, obligándose a comprometer los primeros tres meses en firme (con tres o más meses de anticipación al inicio de la temporada de ventas) y los siguientes meses en forma indicativa con tasas máximas de variación (de la capacidad necesaria) que aumentan a medida que el mes futuro es más lejano. Se resuelve repitiendo el Programa Entero Mixto cada mes (*rolling horizon*).

Los dos artículos de Leung y Ng (2007 a y b) son los únicos que consideran la posibilidad de producir tanto en forma directa como indirecta. El segundo artículo considera incertidumbre en la demanda y por lo tanto plantea un modelo estocástico de dos etapas con posibilidad de cambiar las decisiones pero no permite entregas diferidas (*backorders*).

	1	2	3	4	5	6	7	8	9	10	11	Total
Tareas de planificación	PMP-PCA	PMP	PMP-PD	PMP-PCA-PD	PMP	PMP	PMP-PD	PMP-PCA	PMP-PCA-PD	PMP	PMP	11
Modelo	PLEM	PEM no lineal	Simula-ción	PEM no lineal	PL Posibilista	PEM no lineal	PL Fuzzy	PLEM	PLEM Fuzzy	PLEM Fuzzy	Simula-ción	11
Varios Momts de Dec	-	X	X	-	-	-	X	-	X	-	-	4
Etapas	P-F1-F2	P-F	F-D-Det	F-D-Det-Cls	Fab	Pn-Fn	F-D-ZCls	Fab	P-F-D-C	Fab	P-F	8
Niveles	3	2	1	-	1	N	1	2	N	1	2	6
Costes												
Escasez	X	X	-	X	-	-	-	X	-	-	X	5
Inventario almacenes	X	X	X	X	X	X	X	X	X	X	X	11
Set-up	-	X	-	X	-	X	-	X	-	-	-	4
Producción (variable)	X	X	X	X	X	-	X	X	X	X	X	10
MP (coste de compra)	X	-	-	-	-	-	-	-	X	-	-	2
Transporte	-	-	-	X	-	-	X	-	X	-	X	4
Entregas diferidas	X	-	X	X	X	-	-	-	X	X	X	7
Capacidades												
Plantas (máxima)	X	X	-	X	X	X	X	X	X	X	X	10
Almacenes	X	-	-	X	X	-	X	X	X	-	-	6
Proveedores	X	-	-	-	-	-	-	-	X	-	X	3
Máquinas o líneas	X	-	-	-	X	-	-	X	-	-	-	3
Horas extra	-	-	-	X	X	-	-	X	X	X	-	5
Subcontratación	X	-	-	-	X	-	-	-	-	X	-	3
Tiempo ocioso	-	-	-	X	-	-	-	-	X	-	-	2
Otros												
Modos transporte	-	-	-	X	-	-	-	-	X	-	-	2
Prod directa e indirecta	-	-	-	-	-	-	-	X	-	-	-	1
Aspectos abarcados	12	8	5	12	8	5	7	9	14	6	9	

Tabla 2.14a: Artículos con modelos con incertidumbre

1	Escudero, L.F., Galindo, E., García, G., Gómez, E. y Sabau V. (1999). Schumann, a modelling framework for supply chain management under uncertainty. *European Journal of Operational Research*, 119 (1), 14-34.
2	Barbarosoglu, G. (2000). An integrated supplier-buyer model for improving supply chain coordination. *Production Planning & Control*, 11(8), 732-741.
3	Van Landeghem, H. y Vanmaele, H. (2002). Robust planning: a new paradigm for demand chain planning. *Journal of Operations Management*, 20(6), 769–783.
4	Chen, C.L. y Lee, W.C. (2004). Multi-objective optimization of multi-echelon supply chain networks with uncertain product demands and prices. *Computer & Chemical Engineering*, 28 (6-7), 1131-1144.
5	Wang, R.C. y Liang, T.F. (2005). Applying possibilistic linear programming to aggregate production planning. *Int. Journal of Production Economics*, 98, 328–341.
6	Chen, H. (2006). A Lagrangian relaxation approach for production planning with demand uncertainty. *Proceedings of International Conference on Service Systems and Service Management*, 2, 1020-1025
7	Aliev, R.A., Fazlollahi, B., Guirimov, B.G. y Aliev, R.R. (2007). Fuzzy-genetic approach to aggregate production–distribution planning in supply chain management. *Information Sciences*, 177, 4241–4255.
8	Leung, Stephen C.H. y Ng, Wan-Lung (2007b). A stochastic programming model production planning of perishable products with postponement. Production Plannir & Control, 18(3), 190-202.
9	Peidro, D., Mula, J. y Poler, R. (2007). *Supply chain planning under uncertainty: a fuzzy linear programming approach.* Fuzzy Systems Conference. FUZZ-IEEE 2007. IEEE International. July 2007. Page(s): 1-6.
10	Tavakkoli-Moghaddam, R., Rabbani, M., Gharehgozli, A.H. y Zaerpour, N. (2007). A Fuzzy Aggregate Production Planning Model for Make-to-Stock Environments. Proceedings of the 2007 IEEE IEEM. Page(s): 1609-1613.
11	Díaz-Madroñero, M., Mula, J., Campuzano, F. (2010). Evaluación de proveedores en una cadena de suministro mediante dinámica de sistemas. 4º International Conference on Industrial Engineering and Industrial Management. XIV Congreso de Ingeniería de Organización. Donostia-San Sebastián, España.

Tabla 2.14b: Autores y artículos de la Tabla 2.14a

El modelo de programación matemática propuesto por Peidro et al. (2007) es el que más aspectos abarca y tiene la bondad de considerar varias etapas de la CS, varios niveles, lista de materiales con componentes que pueden ser comunes a distintos productos finales, entregas diferidas, varios medios, costes y tiempos de transporte, y otros aspectos básicos (costes, capacidades, etc.).

2.5. Programación Estocástica bi-etapa

La programación estocástica consiste en optimizar un problema con parámetros inciertos que tienen o no una distribución de probabilidad conocida. Entre las técnicas más utilizadas se encuentra la programación estocástica bi-etapa (*Two-stage stochastic problem*), la cual fue propuesta por Dantzig (1955). Un problema de programación estocástica se puede formular como se muestra en el modelo (2.1). El objetivo, en este caso, es minimizar la suma de los costes de la primera etapa, los cuales son conocidos, y minimizar el valor esperado de los costes de la segunda etapa (Gupta y Maranas, 2003).

$$\begin{aligned} \min \quad & c^T x + p^w q^{wT} y^w \\ s.t. \quad & \\ & Ax = b \\ & -T^w x + W^w y^w = d^w, \quad w \in \Omega \\ & x, \quad y^w \geq 0 \end{aligned} \tag{2.1}$$

En la primera etapa c^T es un vector con los coeficientes de la función objetivo, A es la matriz de coeficientes y b es el vector lado derecho para el conjunto de restricciones asumiendo que hay certidumbre de ellos, y el vector x son las variables de decisión para la primera etapa. En la segunda etapa se introducen y^w como las variables de decisión en función de los escenarios, la matriz tecnológica T^w, la matriz de recursos W^w, el vector lado derecho d^w para cada escenario w, para $w \in \Omega$, donde Ω es el conjunto de escenarios, los coeficientes de la función objetivo q^{wT} y p^w como la probabilidad de ocurrencia de cada w. La utilización de escenarios como una aproximación discreta a funciones de probabilidad, como se muestra en las ecuaciones (2.2) (ver en la siguiente página), es equivalente a la discretización del vector aleatorio continuo como se demuestra en Bakir y Byrne (1998).

$$
\begin{aligned}
&\min z = cx + \theta \\
&\quad s.t. \\
&\qquad -\theta + p^1 q^1 y^1 + \ldots + p^w q^w y^w + \ldots + p^W q^W y^W = 0 \\
&\qquad Ax \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad = b \\
&\qquad -T^1 x + \quad W^1 y^1 \qquad\qquad\qquad\qquad\qquad = d^1 \\
&\qquad\quad \vdots \qquad\quad \ddots \qquad\qquad\qquad\qquad\quad \vdots \\
&\qquad -T^w x \qquad\qquad\quad + W^w y^w \qquad\qquad = d^w \\
&\qquad\quad \vdots \qquad\qquad\qquad\quad \ddots \qquad\quad \vdots \\
&\qquad -T^W x \qquad\qquad\qquad\qquad + W^W y^W = d^W \\
&\qquad\quad x, \qquad y^1, \qquad\qquad y^w, \qquad\qquad y^W \geq 0
\end{aligned}
\tag{2.2}
$$

En programación estocástica hay dos tipos de modelos, los "Espera y observa" (*Wait-See*) y los "Aquí y ahora" (*Here-Now*). Los modelos *Wait-See* (W-S) corresponden a tomar decisiones cuando se ha resuelto la incertidumbre y es equivalente a resolver cada escenario independientemente, mientras que los modelos *Here-Now* (H-N) corresponden a decisiones que se toman sin conocimiento de la variable aleatoria (Bakir y Byrne, 1998). Un modelo de programación estocástica de dos etapas es una combinación de los modelos (H-N) y (W-S), de esta manera se toman decisiones con incertidumbre que sólo afectan la primera etapa y se espera a saber lo que ocurrirá para tomar decisiones sobre la segunda etapa (Gupta y Maranas, 2003).

Dos aplicaciones a la planificación táctica de la producción son:

Escudero et al. (1999) presentan una estructura de modelado para la optimización (minimización del coste total o del coste de entregas diferidas o del coste de pérdida de ventas) del problema de planificación de la cadena de suministro de manufactura, ensamble y distribución (MED) bajo incertidumbre en la demanda del producto, en el coste de aprovisionamiento de los componentes y en el tiempo de entrega. El modelo es un programa lineal multiproducto, multiperiodo y multinivel. Para tratar la incertidumbre usan un análisis de escenarios de dos etapas basado en un enfoque de cambio parcial de plan (*partial recourse*), donde la política de la cadena de suministro MED puede ser implementada para un conjunto inicial de periodos de tiempo, tal que la solución para los demás periodos no necesite ser anticipada (usando el principio de no anticipación) y, luego, esto dependa del escenario para ocurrir. En cualquier caso, se toma en consideración todos los escenarios. Concluyen indicando que

el modelo equivalente determinista para el problema estocástico de dos etapas tiene aún tales dimensiones que es impráctico resolverlo sin usar algún tipo de enfoque de descomposición.

Leung y Ng (2007b) analizan y modelan el proceso de planificación de la producción para productos perecederos con programación estocástica y análisis de escenarios, y agregan finalmente un análisis de sensibilidad al cambio en los costes de escasez. Consideran el aplazamiento (*postponement*) del ensamble final. Su modelo de programación estocástica de dos etapas con cambio de plan no permite entregas diferidas (*backorders*) y penaliza el plan con coste de escasez (*under-fulfillment*). Considera cuatro escenarios (muy bueno, bueno, medio y pobre) de demanda con sus respectivas probabilidades (tres alternativas de diferentes probabilidades). Los costes unitarios de producción y de inventario dependen del escenario y del producto (pero son datos). Los costes de escasez dependen adicionalmente del período (cuanto más cerca al período de mayor demanda, mayores son).

Otros artículos que muestran el uso de la programación estocástica para problemas tácticos de planificación de la CS bajo incertidumbre, son: Pyke y Cohen (1993), Pyke y Cohen (1994), Gupta y Maranas (2000), Von Lanzenauer y Pilz-Glombik (2002), Gupta y Maranas (2003), Lababidi et al. (2004), Miranda y Garrido (2004), Sodhi (2005), Serel (2007), Li y Liu (2008), Patil et al. (2010).

2.6. Conclusiones

Para cada tipo de producto (según sus diferentes características) hay una estrategia de cadena de suministro adecuada de manera que se satisfaga las necesidades del mercado con las condiciones de demanda y de aprovisionamiento particulares.

Para productos innovadores (cuya demanda tiene incertidumbre alta) con proveedor local con coste más bajo que el de los proveedores lejanos, debe emplearse una CS ágil, enfocada en la capacidad de adaptarse a cambios y contratar toda la producción al proveedor local.

Para productos innovadores con múltiples proveedores (al menos un proveedor local con tiempo de aprovisionamiento corto pero coste alto y al menos un proveedor lejano con tiempo de aprovisionamiento largo pero coste bajo), debe

emplearse la estrategia *Base & Surge*. Si los productos no tienen estructura modular, la parte *Base* de la demanda se debe producir con anticipación en una CS ajustada (en un proveedor lejano), y la parte *Surge* de la demanda se debe producir al inicio de la temporada en una CS ágil (en un proveedor local). Si los productos tienen estructura modular, se debe emplear la estrategia *Market Responsive*, y lanzar al mercado un primer lote de productos (producidos contra previsiones en una CS ajustada en un proveedor lejano) y luego se deben atender los pedidos subsiguientes ensamblando contra pedido (en un proveedor local) a partir del stock local de componentes y subensambles (CS agiltada).

Un programa matemático que pretenda hallar el PMP para una CS de productos innovadores debe ser capaz de: abarcar por lo menos un ciclo completo de demanda, planificar en periodos quincenales o menores, abarcar distintos nodos de proveedores considerando sus diferentes capacidades y costes, planificar el uso de la capacidad de producción en dos o más instalaciones, planificar las cantidades a producir de cada uno de los productos finales, subensambles, componentes y partes, en cada nodo y en cada proceso; considerar componentes comunes entre distintos productos finales (modularidad) y materias primas comunes, planificar las cantidades a comprar de cada materia prima en cada nodo, considerar los tiempos de producción y tiempos de transporte desde cada proveedor lejano hasta el país del contratista considerando varias alternativas de transporte, considerar al menos los costes de inventario, entregas diferidas, escasez, producción y transporte, considerar por lo menos las capacidades de las plantas (por proceso), los almacenes y los transportes, considerar la posibilidad de aplazamiento (*postponement*) además de la de producción directa, considerar la incertidumbre de la demanda y la anticipación con la cual se deben colocar los pedidos en firme a los proveedores lejanos y cercanos, y debe poder replanificar la producción (empleando previsiones más actualizadas de la demanda).

De los artículos seleccionados, doce proponen modelos deterministas y diez con incertidumbre. Nueve artículos plantean modelos multiobjetivo con distintas técnicas de resolución. Siete de los artículos consideran la teoría posibilista e incorporan en sus modelos las operaciones de conjuntos difusos tanto para tratar los objetivos como para tratar la incertidumbre de los parámetros.

Ninguno de los modelos encontrados considera a la vez: ciclo de vida corto, incertidumbre alta en la demanda, varias etapas de la CS, varios niveles en la lista de materiales, abastecimiento de los mismos productos desde distintas fuentes con distintos tiempos, medios y costes de transporte, plazos de entrega largos y cortos a la vez, modularidad y posibilidad de aplazar algunos procesos, contratación de la producción con anticipación, cambio de plan de producción en función de nuevas previsiones de la demanda y límites a la capacidad contratada.

Sólo el artículo de Barbarosoglu (2000) considera los requerimientos de los proveedores lejanos para comprometer la producción y contratar la capacidad con anticipación (con límites a los cambios en los periodos subsiguientes), pero no considera proceso de producción directa e indirecta, ni lista de materiales, ni transportes, ni entregas diferidas.

Así mismo, los dos artículos de Leung y Ng (2007 a y b) son los únicos que consideran la posibilidad de producir tanto en forma directa como indirecta. El segundo de sus artículos emplea programación estocástica bi-etapa pero sólo considera al fabricante (no considera proveedores y por ello no tiene envíos entre plantas).

El modelo de programación matemática de Peidro et al. (2007) abarca la lista de materiales, entregas diferidas, tiempo ocioso y varios medios de transporte, pero no incluye los aspectos considerados por Barbarosoglu (2000) (contratar capacidad con anticipación y cambio posterior) y Leung y Ng (2007a y b) (fabricación de productos directos e indirectos).

Estos tres modelos pueden servir de referencia para construir un modelo de programación matemática para la planificación táctica de la producción de productos innovadores.

La programación estocástica ha sido ampliamente empleada para la planificación de la producción en problemas con incertidumbre de la demanda, y la programación estocástica bi-etapa se aplica cuando existen dos o más momentos de decisión.

2.7. Referencias

Bakir, A.M. y Byrne, M.D. (1998). Stochastic linear optimisation of an MPMP production planning model. *International Journal of Production Economics*, 55(1), 87-96.

Barbarosoglu, G. (2000). An integrated supplier-buyer model for improving supply chain coordination. *Production Planning & Control*, 11(8), 732-741.

Blackburn, J.D. (1991). The quick-response movement in the apparel industry: a case study in time compressing supply chains, in Blackburn, J.D. (Ed.), Time Based Competition, Business One Irwin, Homewood, IL, pp. 246-69.

Childerhouse, P., Aitken, J. y Towill, D. (2002). Analysis and Design of Focussed Demand Chains. *Journal of Operations Management*, 20(6), 675-689.

Chopra, S. y Meindl, P. (2001). Supply Chain Management: Strategy, Planning and Operations. 457 pages, Publisher: Prentice Hall; 1st edition, New Jersey.

Christopher, M. y Towill, D. (2001). An integrated model for the design of agile supply chains. *International Journal of Physical Distribution & Logistics Management*, 31(4), 235-246.

Christopher, M., Peck, H. y Towill, D. (2006). A taxonomy for selecting global supply chain strategies. *The International Journal of Logistics Management*, 17(2), 277-287.

Dantzig, G.B. (1955). Linear programming under uncertainty. *Management Science*, 1(3), 197-206.

Escudero, L.F., Galindo, E., García, G., Gómez, E. y Sabau V. (1999). Schumann, a modelling framework for supply chain management under uncertainty. *European Journal of Operational Research*, 119 (1), 14-34.

Fisher, M., Hammond, J., Obermeyer, W. y Raman, A. (1994). Making supply meet demand in an uncertain world. *Harvard Business Review*, 72(3), 83-93.

Fisher, M. (1997). What is the right supply chain for your product? *Harvard Business Review*, 75(2), 105-117.

Fleischmann, B., Meyr, H. y Wagner, M. (2005). Advanced Planning. En Stadtler, H. y Kilger, C. eds., (2005). Supply Chain Management and Advanced Planning. Springer. pp. 81-106.

Gupta, A. y Maranas, C.D. (2000). A two-stage modeling arid solution framework for multisite midterm planning under demand uncertainty. *Industrial & Engineering Chemistry Research*, 39(10), 3799-3813.

Gupta, A. y Maranas, C.D. (2003). Managing demand uncertainty in supply chain planning. *Computers & Chemical Engineering*, 27(8-9), 1219-1227.

Hoekstra, S. y Romme, J. (1992). Integral Logistics Structures: Developing Customer-orientated Goods Flow (New York: Simon and Schuster).

Kumar, S. y Arbi, A.S. (2008). Outsourcing strategies for apparel manufacture: a case study. *Journal of Manufacturing Technology Management*, 19(1), 73-91.

Lababidi, H.M.S., Ahmed, M.A., Alatiqi, I.M. y Al Enzi, A.F. (2004). Optimizing the supply chain of a petrochemical company under uncertain operating and economic conditions. *Industrial & Engineering Chemistry Research*, 43(1), 63-73.

Lamming, R., Johnsen, T., Zheng, J. y Harland, C. (2000). An initial classification of supply networks. International *Journal of Operations and Production Management*, 20 (6), 675–691.

Lee, H.L. (2002). Aligning Supply Chain Strategies with Product Uncertainties. *California Management Review*, 44(3), 105-119.

Leung, S.C.H. y Ng, W-L (2007a). A goal programming model for production planning of perishable products with postponement. *Computers & Industrial Engineering*, 53 (3), 531-541.

Leung, Stephen C.H. y Ng, Wan-Lung (2007b). A stochastic programming model for production planning of perishable products with postponement. *Production Planning & Control*, 18(3), 190-202.

Li, J. y Liu, L. (2008). Supply chain coordination with manufacturer's limited reserve capacity: An extended newsboy problem. *International Journal of Production Economics*, 112(2), 860-868.

Lowson, R. (2001). Analysing the effectiveness of European retail sourcing strategies. European Management Journal, 19(5), 543-551.

Mason-Jones, R., Naylor, J.B. y Towill, D. (2000). Engineering the leagile supply chain. *International Journal of Agile Management Systems*, 2(1), 54–61.

Miranda, P.A. y Garrido, R.A. (2004). Incorporating inventory control decisions into a strategic distribution network design model with strochastic demand. *Transportation Research Part E-Logistics and Transportation Review*. 40(3), 183-207.

Naylor, J.B., Naim, M.M. y Berry, D. (1999). Leagility: Integrating the lean and agile manufacturing paradigms in the total supply chain. *International Journal of Production Economics*, 62(1-2), 107-118.

Olhager, J. (1994). On the positioning of the customer order decoupling point. Proceedings of the Pacific Conference on Manufacturing. Jakarta, 1093-1100.

Patil, R., Avittathur, B. y Shah, J. (2010). Supply chain strategies based on recourse model for very short life cycle products. *International Journal of Production Economics*, 128(1), 3-10.

Peidro, D., Mula, J. y Poler, R. (2007). Supply chain planning under uncertainty: a fuzzy linear programming approach. Fuzzy Systems Conference. FUZZ-IEEE 2007. IEEE International. July 2007. pp. 1-6.

Pyke, D.F. y Cohen, M.A. (1993). Performance-Characteristics of Stochastic Integrated Production Distribution-Systems. *European Journal of Operational Research*, 68(1), 23-48.

Pyke, D.F. y Cohen, M.A. (1994). Multiproduct Integrated Production-Distribution-Systems. *European Journal of Operational Research*, 74(1), 18-49.

Romano, Pietro (2009). How can fluid dynamics help supply chain management? *International Journal of Production Economics*, 118, 463–472.

Serel, D. (2007). Capacity reservation under supply uncertainty. *Computers & Operations Research*, 34(4), 1192-1220.

Shapiro, J. (1998). Bottom-up versus top-down approaches to supply chain modelling. En: Quantitative Models for Supply Chain Management. Kluwer Academic Publishers, Dordrecht, pp. 739–759.

Sodhi, M.S. (2005). Managing demand risk in tactical supply chain planning for a global consumer electronics company. *Production and Operations Management*, 14(1), 69-79.

Stadtler, H. (2005). Supply Chain Management – An Overview. En Stadtler, H. y Kilger, C. eds., (2005). Supply Chain Management and Advanced Planning. 3rd. Edition, Springer, Berlin, pp. 9-35.

Stratton, R. y Warburton, R.D.H. (2003). The strategic integration of agile and lean supply. *International Journal of Production Economics*, 85(2), 183–198

Stratton, R. y Warburton, R.D.H. (2006). Managing the trade-off implications of global supply. *International Journal of Production Economics*, 103(2), 667–679.

Van Landeghem, H. y Vanmaele, H. (2002). Robust planning: a new paradigm for demand chain planning. *Journal of Operations Management*, 20(6), 769–783.

Vonderembse, M.A., Uppal, M., Huang, S.H. y Dismukes, J.P. (2006). Designing supply chains: Towards theory development. *International Journal of Production Economics*, 100(2), 223-238.

Von Lanzenauer, C.H. y Pilz-Glombik, K. (2002). Coordinating supply chain decisions: an optimization model. *Or Spectrum*, 24(1), 59-78.

Wong, C.Y., Arlbjørn, J.S., Hvolby, H-H. y Johansen, J. (2006). Assessing responsiveness of a volatile and seasonal supply chain: A case study. International *Journal of Production Economics*, 104(2), 709-721.

3. DISEÑO DE LOS MODELOS DE PLANIFICACIÓN

3.1. Introducción

Este trabajo de investigación abarca el caso general de las Cadenas de Suministro (CS) integradas por empresas que diseñan productos de temporada y subcontratan la fabricación a distintos proveedores tanto nacionales como extranjeros, y en algunos casos tienen capacidad local propia de producción. Los productos de temporada son innovadores y se ha determinado que la CS de Respuesta Rápida (*Responsive*) es la que mejor los gestiona (Lee (2002), Stratton y Warburton (2003 y 2006), Christopher (2000), Kaipia y Holmström (2007)). Esta estrategia es aplicable a ropa de moda, algunos productos navideños como juguetes e incluso herramientas eléctricas de moda (Wong et al., 2006).

En función del estudio de la bibliografía realizado en el capítulo 2 de esta tesis, se puede afirmar que un modelo matemático para la planificación táctica de las operaciones de CS de Respuesta Rápida deberá necesariamente considerar el aplazamiento (*postponement*) de algunos de los procesos, incluir proveedores de países de bajo coste (y sus condiciones de capacidad y plazos de entrega), tratar la incertidumbre en la demanda, enfrentar los cortos plazos de entrega en la temporada de mayor demanda y abarcar la producción de todos los componentes hasta las materias primas.

El tema de aplazamiento ha sido tratado por distintos investigadores entre los que se puede nombrar a Lee y Billington (1994 y 1995), Garg y Tang (1997), Lee, H.L. (1998), Aviv y Federgruen (2001), Swaminathan y Lee (2003), Gong, H. (2005) y Yang y Yang (2010).

Como se requiere elegir entre diversas alternativas de suministro (proveedores cercanos y lejanos) incluyendo la planificación táctica de la producción de tres o más niveles, con algunos componentes comunes a varios productos terminados, con alta incertidumbre de la demanda cuando se prevé con mucha anticipación y baja incertidumbre si se prevé en el inicio de la temporada de ventas, se ha diseñado un modelo matemático bi-Etapa (que realmente abarca dos modelos y cinco algoritmos) que satisface estos requisitos.

El modelo matemático tiene dos momentos de decisión dentro del horizonte de planificación, por lo tanto el modelo tiene dos fases, para la primera fase se ha diseñado un modelo estocástico en base a escenarios de la demanda, y para la segunda fase, para la que se asume que la demanda es conocida (simulándola con el método de difusión de Bass para productos nuevos), se ha diseñado un modelo determinista que se ejecuta el número de veces que desee el decisor y luego se calcula (con un algoritmo) el promedio de los resultados de todas las ejecuciones**.**

Los modelos diseñados se basan en los trabajos de García-Sabater et al. (2006), Leung y Ng (2007b) y Peidro et al. (2007), de manera que incorporan estructuras de producción alternativas y la posibilidad de hacer aplazamiento, así como la incertidumbre de la demanda, las capacidades de los proveedores y los costes de variación de la capacidad contratada.

El resto del capítulo se organiza como sigue, en el segundo apartado se describe el problema, en el tercer apartado se muestra el proceso de modelado, en el cuarto apartado se presentan los datos del modelo, en el quinto apartado se formulan los modelos matemáticos y se comparan con otros enfoques de modelado, en el sexto apartado se muestra la herramienta de implementación y en el séptimo apartado van las conclusiones.

3.2. Descripción del Problema

El problema general es el de planificación táctica de las operaciones en las CS de Respuesta Rápida (*Responsive*) con estructura alternativa de procesos.

El problema general se concreta en el problema de las empresas que fabrican productos innovadores, cuyo ciclo de producción toma más tiempo que el ciclo de ventas (Harrison y van Hoek, 2008) (ver la figura 3.1).

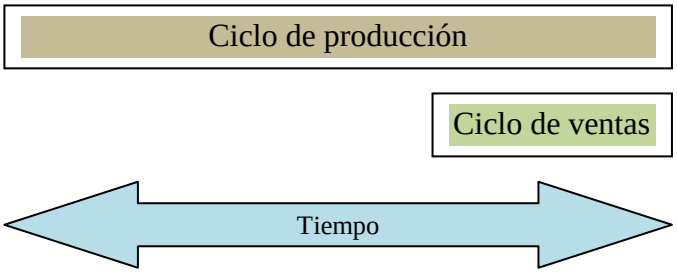


Figura 3.1: Ciclos de producción y ventas (elaboración propia)

Además el problema abarca sólo la parte de la CS conformada por un fabricante local (empresa que puede fabricar los componentes y/o ensamblar el producto final) y uno o varios proveedores en otros países (empresas que pueden fabricar los componentes y/o ensamblar el producto final a menor coste) (ver la figura 3.2). El fabricante local puede subcontratar la producción de componentes y/o de productos finales a uno o varios proveedores. Los proveedores son confiables, es decir cumplen con los plazos de entrega y no fallan en cuanto a la calidad (Lee, 2002), sin embargo, por estar situados en otros países, tienen un plazo de entrega mayor que el de un productor local (Fisher y Raman, 1996) (en el plazo de entrega de los proveedores se incluye el tiempo de transporte hasta un almacén central de productos terminados ubicado en la misma localidad del fabricante local).

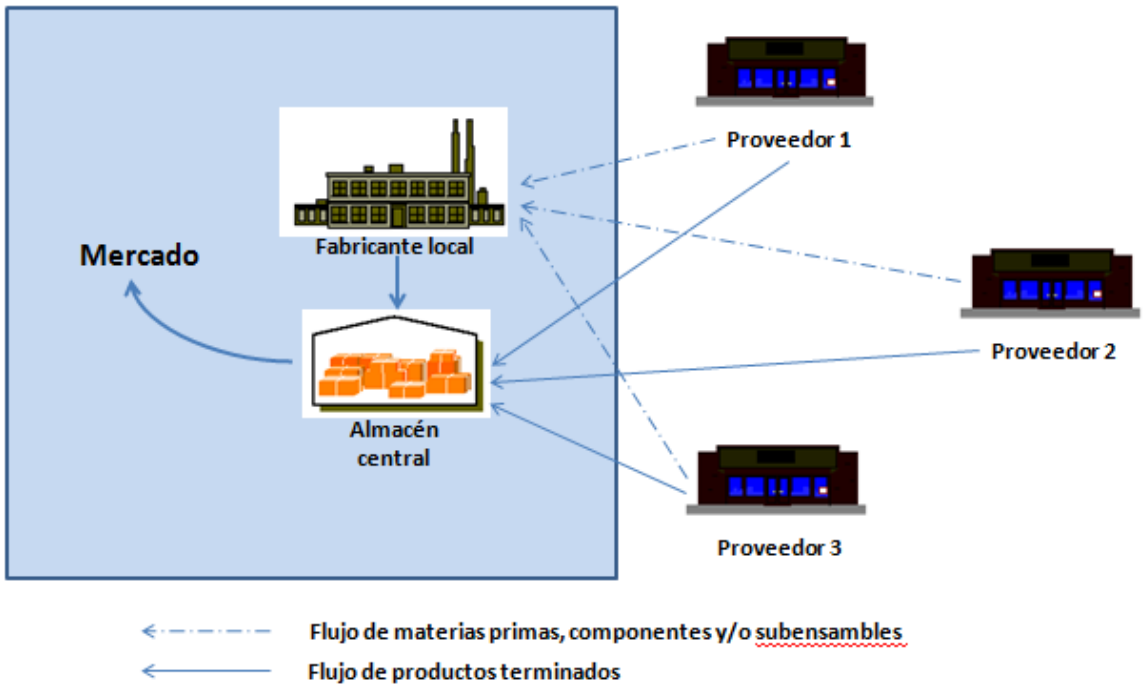


Figura 3.2: Cadena de Suministro de producción (elaboración propia)

Otra característica del problema es que existen dos momentos de decisión para coordinar la producción en la CS (Stratton y Warburton, 2006). En el primer momento de decisión, la capacidad de producción de los proveedores debe ser contratada con varios meses de anticipación al inicio del ciclo de ventas (es decir, cuando aún existe una alta incertidumbre en la previsión de la demanda) (ver la figura 3.3). El segundo momento de decisión es cuando empieza el Ciclo de Ventas (también llamado "temporada de ventas"), que es cuando se puede obtener una previsión muy precisa

(con baja incertidumbre) de la demanda, y en función de esta previsión se definen las cantidades a producir durante el resto de la temporada de ventas (Stratton y Warburton, 2003) (Gutiérrez, 2007).

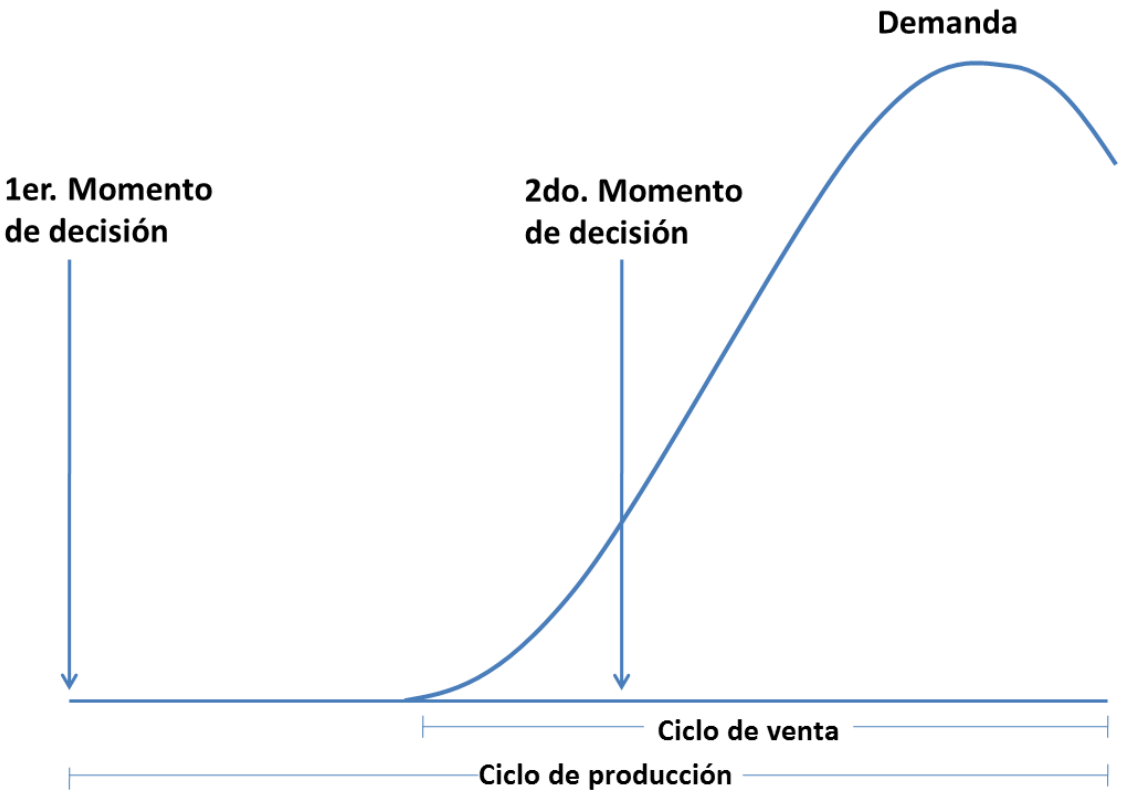


Figura 3.3: Momentos de decisión (elaboración propia)

Las principales características del problema son:

1. Productos innovadores.
2. Demanda impredecible y volátil. Demanda concentrada en pocos meses del año.
3. Diferencias entre la producción y la demanda por exceso y defecto.
4. Precio de venta fijo y valor residual (precio de remate) fijo de los productos al final de la temporada de ventas.
5. Muchos productos diferentes y varios niveles en su lista de materiales.
6. Productos con diseño modular con componentes comunes.
7. Las materias primas se pueden comprar tanto localmente como en el extranjero con distintos costes.
8. Existencia de empresas de producción local y de producción en el extranjero (Proveedores), diferentes en costes, capacidades y plazos de entrega. Con diferentes procesos.

9. En las empresas de producción lejanas se contrata la Capacidad de Producción con tres o más meses de anticipación al inicio del Ciclo de Ventas.
10. Sobrecostes de las empresas de producción lejanas cuando se sobrepasa la capacidad contratada en un periodo dado.
11. Sobrecostes de las empresas de producción locales cuando se sobrepasa la capacidad instalada en un periodo dado, y costes de sub-utilización cuando no se aprovecha toda la capacidad instalada (tiempo ocioso).
12. Varios medios de transporte con costes, tiempos y capacidades diferentes.

3.2.1. Descripción de las características del problema

1. **Productos innovadores:** son productos nuevos para el mercado (pueden ser pequeñas innovaciones de productos maduros como en el caso de los juguetes y la ropa de moda), con ciclo de vida corto (normalmente pocos meses), cuya característica ganadora (*order winner*) es la novedad y la disponibilidad, muy seguida del coste (Romano, 2009).

2. **Demanda impredecible y volátil:** el problema está enfocado en los productos innovadores, para los cuales la certidumbre de la previsión de la demanda es menor a medida que la anticipación es mayor (Christopher et al., 2006). Según Blackburn (1991), los errores de previsión en la industria de la moda en función del horizonte de tiempo, a "ojo de buen cubero" (*rule of thumb* en el original), son:

    - Al inicio de la estación ± 10 por ciento,
    - 16 semanas antes ± 20 por ciento, y
    - 26 semanas antes ± 40 por ciento.

    El margen promedio de error en la previsión de productos innovadores cuando se tiene que contratar la capacidad (primer momento de decisión) es de 40% a 100% (Fisher, 1997) (Wong et al., 2006).

    Además la demanda es volátil porque desaparece al final de la temporada de ventas (que coincide con el fin del ciclo de vida del producto). La demanda es altamente estacional, concentrada en pocos meses del año. El coste de obsolescencia es alto y la tasa de obsolescencia también es muy alta (Kurawarwala y Matsuo, 1996).

Sólo los productos terminados tienen demanda independiente (demanda externa). Los requerimientos de subensambles, componentes, partes y materias primas están en función de las necesidades de productos terminados, por lo tanto aquellos tienen demanda dependiente (demanda interna) (Leung y Ng, 2007).

3. **Diferencias entre la producción y la demanda por exceso y defecto:** cuando la producción acumulada es mayor que la demanda acumulada en cada periodo, se generan inventarios y éstos se penalizan con un coste de inventario; en cambio cuando la producción acumulada es menor que la demanda acumulada en cada periodo, se tienen Pedidos Pendientes (sólo de productos terminados) y éstos se penalizan con costes de pedidos pendientes (Guiffrida y Nagi, 2006). Los pedidos pendientes de un periodo pueden ser atendidos con la producción del periodo siguiente, por ello también se les denomina "entregas diferidas" (en inglés: *backorders*).

   Al final de la temporada de ventas los costes de pedidos pendientes se denominan costes de escasez. Fisher (1997) afirma que la tasa promedio de escasez es de 10% a 40% al final de la temporada de ventas para los productos innovadores.

4. **Precio de venta fijo y valor residual (precio de remate) de los productos al final de la temporada de venta:** el precio de venta es fijo durante toda la temporada de ventas y, cuando la producción acumulada de un producto es mayor que su demanda acumulada al final de la temporada (es decir, cuando sobra producto), se vende el producto sobrante a un precio de remate al cual llamamos Valor Residual (Kisperska-Moron y Swierczek, 2011). No todo el producto sobrante se puede vender a su valor residual, sólo un determinado porcentaje de la demanda total (por ejemplo un 10%) se podrá rematar al valor residual. Si hubiera más producto sobrante que el porcentaje citado, el exceso se remataría a un valor menor que el valor residual (por ejemplo 50% del valor residual).

5. **Muchos productos y varios niveles en su lista de materiales:** el fabricante ofrece al mercado, en cada temporada, muchos productos terminados, cada uno tiene niveles de: subensambles, componentes y materias primas. La lista de materiales contiene las cantidades de cada subensamble, componente y materia

prima necesarias para la producción de cada producto terminado (Chern y Hsieh, 2007).

6. **Productos con diseño modular con componentes comunes:** los productos tienen diseño modular y, además, hay componentes y materias primas que son comunes a varios productos terminados. Esto permite desacoplar la fabricación de los componentes de la ejecución del ensamble final, de manera que la producción de componentes se puede adelantar y almacenar, para luego ensamblar los productos en función de previsiones más precisas (Kisperska-Moron y Swierczek, 2011). Sin embargo también se puede fabricar un producto de principio a fin en forma ininterrumpida (incluyendo la fabricación de los componentes) (Leung y Ng, 2007).

7. **Las materias primas se pueden comprar tanto localmente como en el extranjero con distintos costes:** las compras locales tienen un plazo de entrega corto y coste alto mientras las compras de proveedores lejanos (de ultramar) tienen plazo de entrega largo y coste bajo (Kumar y Arbi, 2008). Stratton y Warburton (2006) indican que el plazo de entrega para suministro global de hilo por barco es de 4 semanas.

8. **Existencia de empresas de producción locales y en el extranjero (Proveedores), diferentes en costes, capacidades y plazos de entrega:** las empresas de producción locales tienen mayor coste de mano de obra, una determinada capacidad máxima y plazos de entrega cortos, mientras las empresas de producción en países de costes bajos tienen menor coste de mano de obra, una capacidad máxima diferente y plazos de entrega largos. La gran diferencia está en los costes de mano de obra (1 a 10 según Kumar y Arbi, 2008), costes de materias primas y otros costes locales; en cambio cuando un proceso es automatizado y requiere poca mano de obra casi no hay diferencia en el coste de producción. Las empresas de producción en países de costes bajos tienen plazos de entrega largos porque tienen que incluir en sus plazos de entrega el tiempo de transporte desde su país hasta el país del contratante. Si bien los plazos de entrega son diferentes, las empresas de producción tanto local como en el extranjero tienen procesos estables (a diferencia de los procesos “en evolución” -Lee, 2002-) y por ello la incertidumbre en el aprovisionamiento es nula. El

número y localización de los Proveedores (empresas en el extranjero) es conocido.

9. **En las empresas de producción lejanas se contrata la capacidad de producción con tres o más meses de anticipación al inicio del ciclo de ventas:** dado que el Ciclo de Producción es mayor que el Ciclo de Ventas, la capacidad de producción de los proveedores lejanos se contrata con la anticipación suficiente para abarcar su plazo de entrega y que los productos lleguen a tiempo al mercado (Kaipia y Holmström, 2007). El fabricante local, antes del inicio del ciclo de producción, calcula sus necesidades de producción en función de sus previsiones de la demanda y, contrata en firme la capacidad necesaria de los proveedores lejanos para todo o parte del ciclo de producción. Esta capacidad contratada no se puede cambiar después, pero sí se puede sobre-utilizar (pagando un coste adicional) o sub-utilizar (sin coste adicional).

10. **Sobrecostes de las empresas de producción lejanas cuando se sobrepasa la Capacidad Contratada en un periodo dado:** existen costes de sobre-utilización de la capacidad contratada al proveedor lejano (Boulaksil et al., 2011) cuando en algún periodo se requiere más capacidad que la contratada previamente; así mismo, si no se utiliza la capacidad contratada se tiene el coste de oportunidad por no aprovechar esa capacidad en un periodo dado (es decir, la capacidad contratada se tiene que pagar aunque no se use). La sobre-utilización de la capacidad contratada tiene un límite superior que depende que la capacidad contratada en horas normales (ver la figura 3.4).

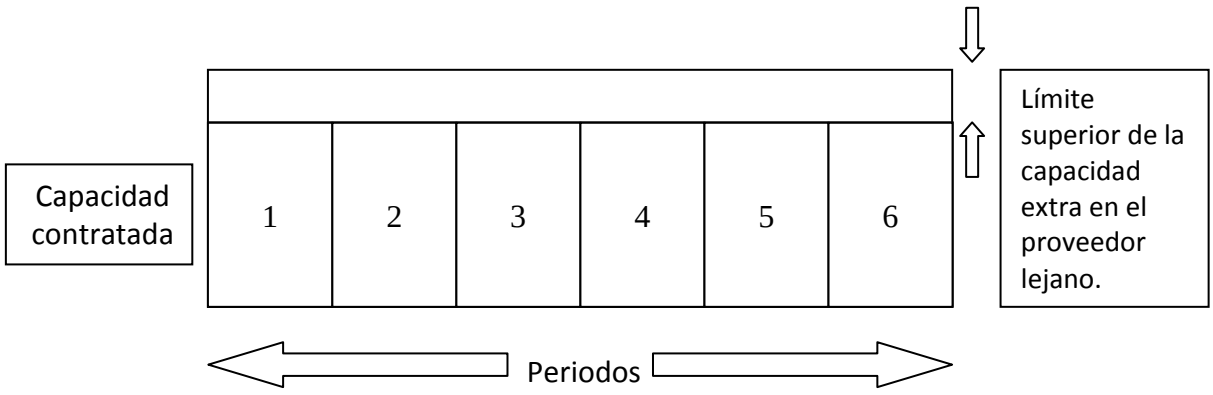


Figura 3.4: Capacidad contratada y capacidad en horas extra (elaboración propia)

11. **Sobrecostes de las empresas de producción locales cuando se sobrepasa la Capacidad Máxima en un periodo dado:** los proveedores locales tienen una determinada capacidad instalada a la cual llamamos "Capacidad Máxima"; si en un periodo dado se exige producir más de lo que se puede hacer en su capacidad instalada en turno normal, aparecen costes de sobre-utilización de su capacidad máxima (horas extra). Por lo contrario, si no se aprovecha la capacidad máxima, aparecen costes de sub-utilización (Christopher, 2000).

12. **Varios medios de transportes con costes, tiempos y capacidades diferentes:** se emplean básicamente dos medios de transporte entre las empresas de ultramar y las empresas locales: barco y avión. El barco tiene menor coste pero más capacidad y mayor tiempo. El coste del transporte en barco es de 1 por ciento del valor del producto (Guiffrida y Nagi, 2006). El avión tiene mayor coste pero menos capacidad y menos tiempo.

En la Tabla 3.1 se presenta una visión sinóptica del problema.

	Características
Alcance	Planificación táctica de las operaciones en las Cadenas de Suministro de Respuesta Rápida con estructura alternativa de procesos.
Etapas de la CS	Proveedores, fabricante y almacén central.
Productos terminados (pt)	Innovadores.
Demanda externa	Estacional y sólo de pt. La precisión de la previsión es menor a medida que la anticipación es mayor.
Precio de venta de los pt	Precio fijo y conocido para cada pt.
Componentes y materias primas	Varios y compartidos según la lista de materiales, con un número de niveles conocido.

Procesos	Los proveedores lejanos y locales tienen los procesos necesarios para la producción. Cada proceso tiene diferente capacidad.
Aplazamiento	Se puede adelantar algunos procesos y aplazar el ensamble final para los periodos de mayor demanda.
Proveedores	Todos los proveedores son conocidos y se conocen todos sus datos.
Plazo de entrega	Es el tiempo total desde que se compromete la producción hasta que llegan los componentes al fabricante local o los pt al almacén central.
Tiempos de producción	Tiempos de fabricación por componente, subensamble y pt, por proceso y por proveedor.
Capacidad de producción	Capacidad máxima de cada proveedor por proceso (es dato) en horas normales y en horas extra.
Contratación de capacidad	La capacidad de los proveedores lejanos se contrata para cada proceso en cada periodo.
Costes de producción	Dependen del proveedor y son fijos y conocidos (incluyen los impuestos de importación y costes de cambio de moneda), y pueden ser diferentes en los distintos periodos.
Costes de sobre-utilización	Coste de exceder la capacidad contratada a un proveedor lejano y coste de exceder la capacidad máxima de un proveedor local.

Costes de sub-utilización	Coste de no usar la capacidad máxima.
Coste de las materias primas	Depende del país del proveedor pero es fijo y conocido.
Costes de transporte	Depende del medio usado, origen y destino, pero son fijos y conocidos para cada materia prima, componente, subensamble y pt.
Costes de inventario	Costes de almacenamiento por proveedor, por ítem y por periodo.
Coste de pedidos pendientes	Costes por unidad, fijos y conocidos pero aumentan hacia el mes de mayor demanda.
Coste de escasez	Coste al final de la temporada por ventas perdidas.
Valor residual	Precio de remate del producto no vendido al final de la temporada.
Medios de transporte	Dos o tres medios de transporte (avión, barco, etc.).
Capacidad de transporte	Capacidad máxima por cada medio y periodo.
Tiempo de transporte	Depende del medio y de la distancia del origen al destino.

Tabla 3.1: Características del problema en estudio (elaboración propia)

Todos los costes y todos los límites de capacidad son datos conocidos pero pueden ser diferentes en cada periodo de tiempo. En este problema se considera que el grado de coordinación entre los proveedores lejanos y el fabricante local (que es el planificador central) es elevado y por ello se conocen todos sus costes, procesos, capacidades y tiempos de producción (Lejeune, 2005).

3.3. Propuesta de Modelado

Este apartado abarca la Planificación Táctica de las Operaciones de las CS integradas por empresas que fabrican productos innovadores y subcontratan parte o toda la producción a distintos proveedores tanto locales como extranjeros.

En función del estudio de la bibliografía realizado en el capítulo 2 de esta tesis se puede afirmar que un modelo matemático para la planificación táctica de CS de Respuesta Rápida deberá necesariamente considerar el aplazamiento (*postponement*) de algunos de los procesos, incluir proveedores de países de bajo coste (y sus condiciones de capacidad y plazos de entrega), tratar la incertidumbre en la demanda, tener dos o más momentos de decisión y abarcar dos o más niveles de componentes (que pueden ser compartidos o no entre los diferentes productos). Este es el caso típico de la planificación táctica de la producción con dos etapas y cambio de plan como lo definen Escudero et al. (1999) y Leung y Ng (2007b).

Según Rohde y Wagner (2005) un modelo de planificación de la producción debe abarcar un ciclo completo para ser capaz de balancear todos los picos estacionales. En el presente caso de estudio, un ciclo completo es menor a un año pues se tienen productos innovadores con un solo pico estacional. Se plantea el modelo de manera que se empiece con suficiente anticipación para producir todos los productos necesarios para satisfacer la demanda y se termine de producir en el mes en el que acaba la demanda o antes.

El objetivo de este apartado es desarrollar un modelo matemático que permita analizar la planificación táctica de las operaciones en una CS de Respuesta Rápida, para determinar en qué condiciones conviene contratar la producción a proveedores locales frente a proveedores lejanos o a ambos en distintos periodos y cantidades, y además determinar cómo afectan los resultados: distintos grados de incertidumbre de la demanda, distintos valores residuales del inventario final, distintos costes de sub-utilización, distintos momentos de re-planificación, distintas capacidades y distintos costes de pedidos pendientes.

Primeramente se establece que el modelo debe determinar:

- Plan de producción de cada proveedor (local y extranjero).

- Capacidad a contratar de cada proveedor extranjero.
- Cantidad de ventas e ingresos.
- Nivel de inventario de cada empresa y de entregas diferidas de pt.
- Los costes de producción, inventario, escasez, sobre-utilización y sub-utilización del fabricante local y de cada uno de sus proveedores.
- Nivel de servicio (grado de satisfacción puntual de la demanda).
- Y debe maximizar el beneficio (margen bruto: ingresos menos costes).

A continuación se analiza cada una de las características del problema real y se plantea la forma de tratarlas en el modelo matemático.

1. **Productos innovadores:** dado que su *order winner* es la disponibilidad seguida del coste, la función objetivo del modelo debe lograr una alta disponibilidad al menor coste posible. Esto se consigue incluyendo en la función objetivo del modelo matemático los costes de pedidos pendientes de cada producto en cada periodo (Lee et al., 2002). Se debe asignar un coste unitario elevado a los pedidos pendientes de manera que al buscar minimizar el coste total se incurra lo menos posible en pedidos pendientes. Sólo los productos con demanda externa deben poder tener pedidos pendientes y su coste debe ser mayor a medida que se adentra en la temporada de ventas. Este coste unitario de pedidos pendientes es un coste de oportunidad que cada empresa debe definir por producto y por periodo. Como el objetivo del modelo es maximizar el beneficio, en la función objetivo se incluyen las ventas menos todos los costes (producción, inventario, etc.), de manera que se consiga la relación entre ventas y costes que da el mayor beneficio (Jackson y Grossmann, 2003).

2. **Demanda impredecible y volátil:** dado que la demanda está casi totalmente centrada en una temporada de ventas, la precisión de las previsiones de la demanda es menor a medida que la anticipación es mayor y, además, el ciclo de producción es mayor que el ciclo de ventas: el modelo considera dos momentos de decisión dentro del horizonte de planificación (es decir, dentro del ciclo de producción) (Escudero et al., 1999). En el primer momento de decisión, con

mucha anticipación al inicio de la temporada de ventas, el modelo "Fase I" partirá de previsiones la demanda mediante escenarios, asociando una probabilidad de ocurrencia a cada escenario. Estos escenarios y su probabilidad serán datos para el modelo, datos generados por expertos en cada aplicación del modelo (ver la figura 3.5). Este modelo determinará a qué proveedores contratar, la capacidad a contratar de los proveedores lejanos y la producción a asignar a cada proveedor (locales y lejanos) en cada periodo para todo el horizonte de planificación.

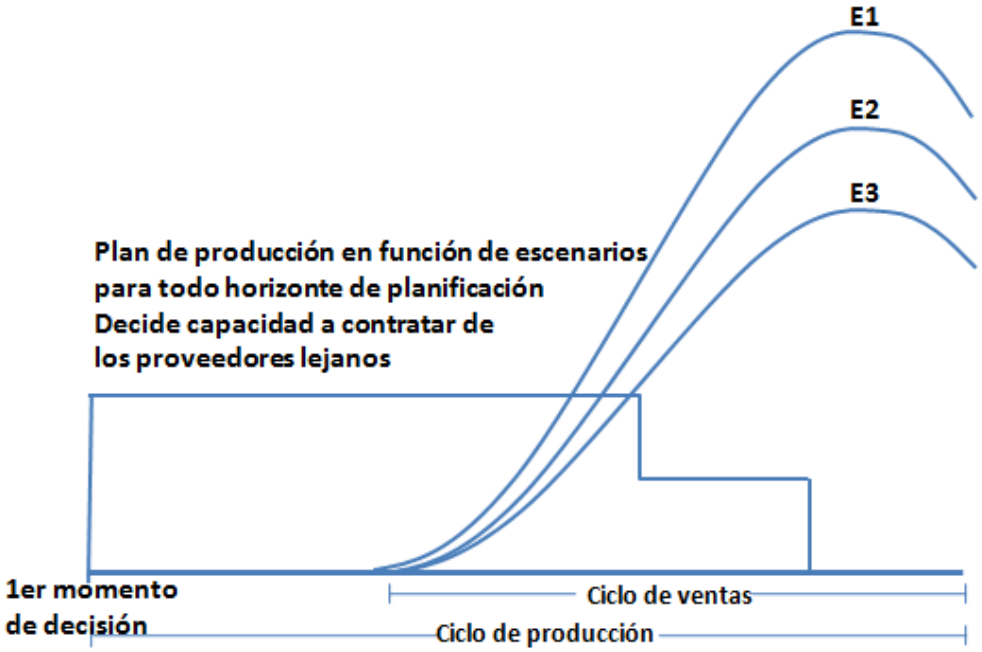


Figura 3.5: Primer momento de decisión (elaboración propia)

Luego, en el segundo momento de decisión, una vez iniciada la temporada de ventas (ver la figura 3.6 en la siguiente página), el modelo "Fase II" partirá de la demanda prevista por el método de Difusión de Bass (por ser éste adecuado para prever la demanda de productos nuevos). En el momento de comenzar la segunda etapa, se tendrá una buena estimación de los parámetros (m, p y q) que usa el método de difusión de Bass (Mahajan et al., 1995). En la práctica real, los usuarios deberán elegir su propio procedimiento de previsión. El modelo "Fase II" re-planificará la producción a partir del segundo momento de decisión, tanto de los proveedores locales como de los lejanos, respetando la capacidad contratada de los proveedores lejanos y la capacidad máxima de los proveedores locales.

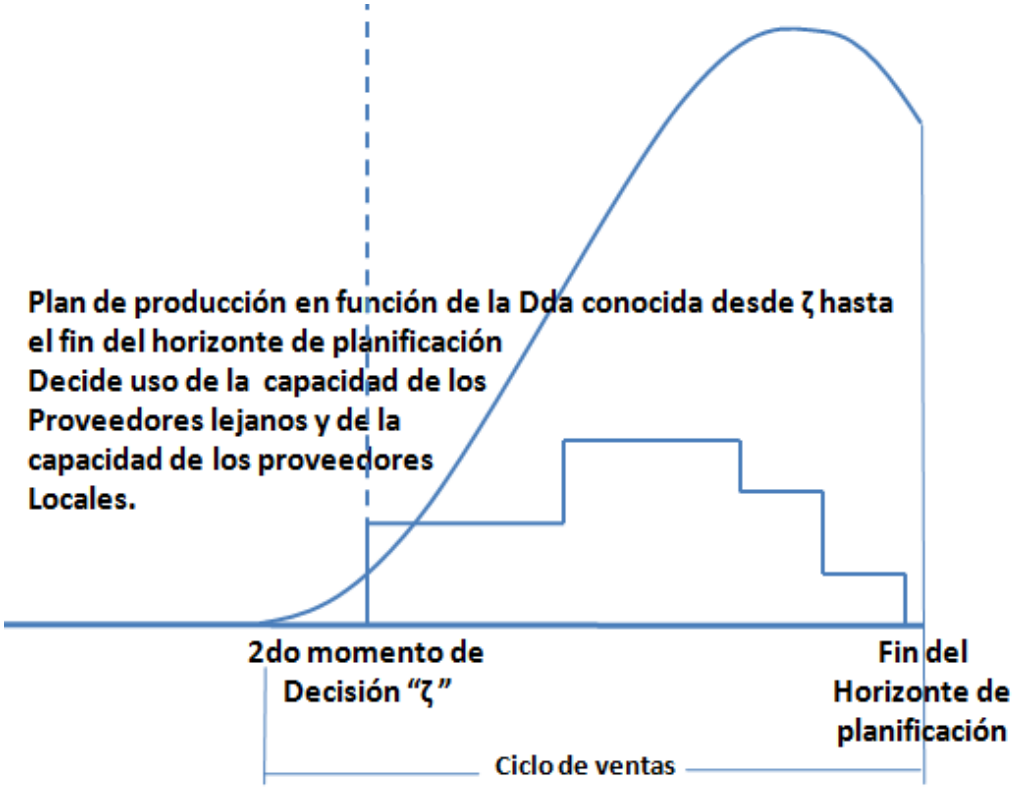


Figura 3.6: Segundo momento de decisión (elaboración propia)

3. **Diferencias entre la producción y la demanda por exceso y defecto:** dado que sólo los productos terminados pueden tener una producción menor que la demanda, el modelo incluye una variable de ventas para que sea la parte de la demanda que realmente satisface la producción en cada periodo. Cuando las ventas son menores la demanda, se generan *"*pedidos pendientes*"* (*backorders*) y éstos son acumulables en el tiempo (es decir, se agregan a la demanda del siguiente periodo). Entonces, la relación entre las ventas y la demanda se plantea así: los pedidos pendientes ($\boldsymbol{\beta_t}$) de un producto terminado en un periodo *t* son iguales a los pedidos pendientes del mismo producto en el periodo anterior ($\boldsymbol{\beta_{t-1}}$) más la demanda ($\boldsymbol{D_t}$) de ese producto terminado en el periodo *t* menos las ventas del mismo periodo ($\boldsymbol{V_t}$). Las variables β_t y V_t sólo pueden tener valores mayores o iguales a cero (Peidro et al., 2007).

$$\beta_{it} = \beta_{i,t-1} + D_{it} - V_{it} \qquad \text{para } i\text{=productos terminados, } \forall\, t$$

En el último periodo del horizonte de planificación, los pedidos pendientes equivalen a las ventas perdidas y tienen un coste unitario de escasez que es

conocido (Baykasoglu, 2001). En el modelo, el coste de los pedidos pendientes aumenta a medida que se aproxima el final del horizonte de planificación (Leung y Ng, 2007a).

4. **Precio de venta fijo y valor residual (precio de remate) de los productos al final de la temporada de venta:** el precio de venta y el valor residual de cada producto son datos para el modelo y se incluyen en su función objetivo de manera que ésta considere los ingresos totales por ventas y los ingresos por los productos sobrantes a su precio de remate (valor residual). Como los productos terminados que queden en almacén serán los artículos no vendidos y tendrán que ser rematados a un precio más bajo, el modelo deberá permitir aumentar el coste de inventario en el último periodo (esto hará que el modelo matemático minimice el inventario sobrante).

5. **Muchos productos y varios niveles en su lista de materiales:** el modelo considera que se parte de materias primas y se pueden producir componentes, subensambles y productos terminados en cada proveedor. El modelo considera la lista de materiales para saber las cantidades de cada subensamble, componente y materia prima necesarias para la producción de cada producto terminado (Arntzen et al., 1995). Se modela los procesos usando el concepto de los *strokes* originalmente propuesto por Garcia-Sabater et al., (2006) y posteriormente utilizado en el contexto de CS en Calderón-Lama et al., (2009). El *stroke* es la representación de una actividad a ejecutar, que puede ser de conformación y/o subensamble y/o ensamble final. El *stroke* se puede entender como el proceso de transformación de un lote X en un lote Y, por lo tanto es el *stroke* quien consume recursos en la generación del producto; con este concepto se planifican los procesos (los *strokes*) y no los materiales. El *stroke* consume una cantidad determinada de un o unos ítems (materias primas, componentes o subensambles) y produce otra cantidad determinada de un ítem de nivel superior en su lista de materiales. Las cantidades que consume y produce cada *stroke* son datos para el modelo y dependen de cada aplicación.

Por ejemplo: si para producir una unidad de A se requiere una unidad de B, y ésta a su vez requiere una unidad de C, y C requiere una unidad de la materia

prima D (que se compra), mediante los *strokes* se pueden modelar cuatro formas de llegar al producto terminado A (ver la figura 3.7):

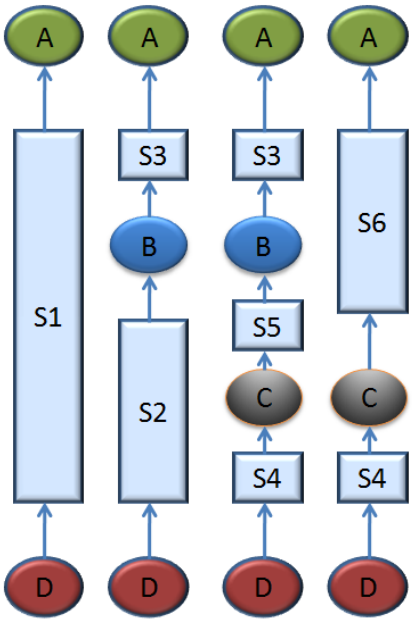


Figura 3.7: Cuatro formas de producir A (elaboración propia)

1) el *stroke* 1 (S1) parte de la materia prima D y entrega el producto terminado A (incluye varios procesos pero sin almacenamiento intermedio y se planifican como un único proceso);

2) el *stroke* 2 (S2) consume la materia prima D y produce el subensamble B (este ítem se almacena) y luego el *stroke* 3 (S3) toma el subensamble B y los transforma en A (el S3 abarca diferentes procesos que el S2);

3) el *stroke* 4 (S4) toma D y la convierte en C, el cual se almacena. El *stroke* 5 (S5) toma C y lo transforma en B, que también se almacena. El *stroke* 3 (ya mencionado antes) toma B y lo convierte en A;

4) El *stroke* 4 transforma D en C y se almacena. Luego el *stroke* 6 (S6) toma C y lo convierte en A (en un proceso sin almacenamiento intermedio).

La información de los ítems (y su cantidad) que consume cada *stroke* se ingresa mediante tablas como la tabla 3.2.

M(i,k)	Stroke					
SKU	S1	S2	S3	S4	S5	S6
A	0	0	0	0	0	0
B	0	0	1	0	0	0
C	0	0	0	0	1	1
D	1	1	0	1	0	0

Tabla 3.2: Lista de materiales que consume cada *stroke*.

De igual manera se ingresa la información de los ítems que genera cada *stroke* (ver la tabla 3.3).

N(i,k)	Stroke					
SKU	S1	S2	S3	S4	S5	S6
A	1	0	1	0	0	1
B	0	1	0	0	1	0
C	0	0	0	1	0	0
D	0	0	0	0	0	0

Tabla 3.3: Lista de ítems que genera cada *stroke*.

6. **Productos con diseño modular con componentes comunes:** el modelo matemático, modelado mediante *strokes*, permite planificar el uso de componentes comunes en distintos productos terminados. Por ejemplo, el *stroke* 7 (S7) toma el ítem B (ver la figura 3.8) y lo transformar en ítem E (diferente del ítem A), porque el S7 es un proceso diferente del S3. Los *strokes* permiten desacoplar la fabricación de los componentes de la ejecución del ensamble final, de manera que la producción de componentes se puede adelantar y almacenar, para luego ensamblar los productos en función de previsiones más precisas. Sin embargo, también se puede fabricar un producto de principio a fin en forma ininterrumpida (Leung y Ng, 2007a), por ejemplo el *stroke* 1 (S1).

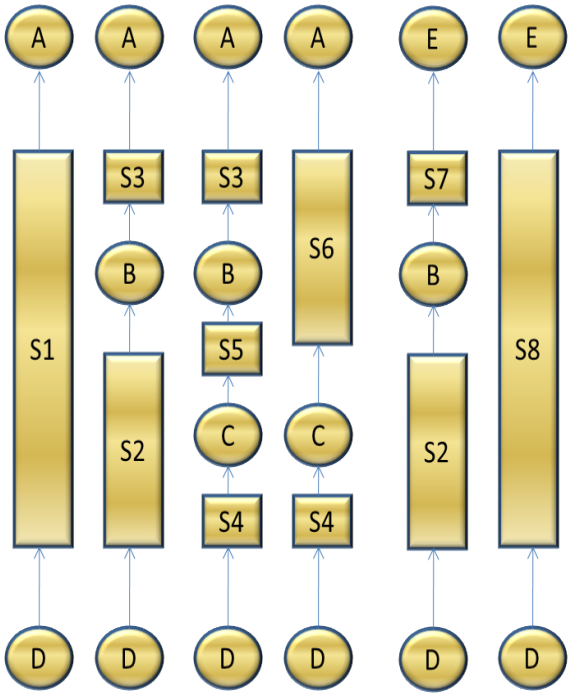


Figura 3.8: Componentes comunes y procesos diferentes (elaboración propia)

7. **Las materias primas se pueden comprar tanto localmente como en el extranjero con distintos costes:** la materia prima la puede comprar cada proveedor en su propio país (Lee et al., 2002). Los *strokes* permiten tomar ítems de un país y entregar otros ítems en el mismo u otro país, por ejemplo, el *stroke* 8 (S8) toma materia prima D de China, la transforma y transporta, y entrega producto terminado E en España (ver la figura 3.8). Los *strokes* pueden abarcar uno o más procesos de un proveedor y abarcar los transportes. La denominación que se dé al ítem que resulta de cada *stroke* no sólo sirve para identificarlo sino que también sirve para diferenciarlo por su ubicación, por ejemplo en la figura 3.8 el ítem B puede ser igual al ítem C, sólo que B está en España y C está en China, en este caso el *stroke* 5 (S5) es el transporte de China a España. Los *strokes* también modelan el envío de ítems de un proveedor a otro.

   Para indicar que ítems se pueden comprar y cuáles no, se emplea un parámetro denominado "Autoriza Compra" (AC) que puede valer 1 o 0 según permita la compra del ítem en cada periodo o no lo permita, respectivamente. Estos valores se introducen como dato para cada ítem en cada periodo ($AC_{i,t}$).

8. **Existencia de empresas de producción locales y en el extranjero (proveedores), diferentes en costes, capacidades y plazos de entrega:** en el modelo se tendrán diferentes *strokes* para cada una de las empresas de producción. Los *strokes* tienen asociado un tiempo de ejecución llamado "plazo de entrega" (*lead time*), así mismo cada *stroke* tiene un coste asociado y consume una determinada cantidad de uno o varios recursos de cada proveedor. Los *strokes* asumen los plazos de entrega y los costes pero las capacidades máximas dependen de cada proveedor y son dato para el modelo. Los *strokes* consumen recursos (máquinas, mano de obra, etc.) de cada proveedor; la cantidad de recursos que consume cada *stroke* por periodo es dato para el modelo.

   La información de los recursos (y su cantidad) que consume cada *stroke* se ingresa mediante tablas como la tabla 3.4. Nótese que los tres primeros recursos pertenecen al proveedor lejano y el recurso R3e pertenece al proveedor local.

Coeficientes tecnológicos	Stroke					
Recursos	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**
R1	1	1	0	1	0	0
R2	1	1	0	0	1	1
R3	1	0	0	0	0	1
R3e	0	0	1	0	0	0

Tabla 3.4: Lista de recursos que consume cada *stroke*.

El modelo determina la capacidad usada de cada recurso multiplicando los *strokes* ejecutados por su consumo de recursos. En el primer momento de decisión, el modelo decide los *strokes* a ejecutar y por lo tanto la capacidad a usar, de manera que ésta siempre sea menor o igual a la capacidad máxima del correspondiente recurso en el correspondiente proveedor (Calderón-Lama et al., 2009). Se contrata la capacidad de los proveedores lejanos que resulte necesaria y ésta se paga completamente, de manera que los *strokes* que ejecutan los proveedores lejanos no se costean, sólo se considera su tiempo y uso de recursos. En cambio, los *strokes* que ejecutan los proveedores locales sí tienen coste, tiempo y uso de recursos, así la capacidad local se paga al ejecutar los *strokes*, esto es equivalente a pagar por la producción realizada.

En la tabla 3.5 se muestra un ejemplo de ingreso de costes de los *strokes*. Se puede ver que sólo el *stroke* tres tiene coste porque es el único que hace el proveedor local. Estos costes pueden variar de un periodo a otro.

Coste Stroke	Periodo					
Stroke	**1**	**2**	**3**	**4**	**5**	**6**
S1	0	0	0	0	0	0
S2	0	0	0	0	0	0
S3	15	15	15	15	15	15
S4	0	0	0	0	0	0
S5	0	0	0	0	0	0
S6	0	0	0	0	0	0

Tabla 3.5: Coste unitario de los *strokes*.

En la tabla 3.6 se muestra un ejemplo de ingreso de costes unitarios de los recursos. Se puede ver que el recurso R3e no tiene coste por ser del proveedor local. Estos costes pueden ser diferentes en cada periodo.

Coste Capacidad	Periodo					
Recurso	**1**	**2**	**3**	**4**	**5**	**6**
R1	4	4	4	4	4	4
R2	0.25	0.25	0.25	0.25	0.25	0.25
R3	5	5	5	5	5	5
R3e	0	0	0	0	0	0

Tabla 3.6: Coste unitario de los recursos.

En la tabla 3.7 se muestra un ejemplo de plazos de entrega de los *strokes* (medidos en periodos de planificación).

Lead time	Stroke					
Plazo de entrega	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**
	3	2	1	1	1	2

Tabla 3.7: Plazo de entrega de cada *stroke*.

En la tabla 3.8 se muestra un ejemplo de capacidad máxima de cada recurso (en miles de unidades) por periodo.

Capacidad	Periodo					
Recurso	**1**	**2**	**3**	**4**	**5**	**6**
R1	20	20	20	20	20	20
R2	100	100	100	100	100	100
R3	20	20	20	20	20	20
R3e	10	10	10	10	10	10

Tabla 3.8: Capacidad de los recursos.

En el punto 11 se explican los sobrecostes por sobre-utilización y sub-utilización de la capacidad de los proveedores locales.

9. **En las empresas de producción lejanas se contrata la capacidad de producción con tres o más meses de anticipación al inicio del ciclo de ventas:** dado que el ciclo de producción es mayor que el ciclo de ventas, el modelo decide la capacidad a contratar de los proveedores lejanos en el primer momento de decisión. Para esto el modelo debe tener dos fases (Bakir y Byrne, 1998); en la primera fase el modelo se ejecuta contra previsiones de la demanda muy anticipadas (con escenarios) (Escudero et al. 1999), este modelo estocástico abarca todo el horizonte de planificación, planifica la producción y decide las

capacidades a contratar de cada uno de los proveedores lejanos. Esta decisión se mantiene fija en la segunda fase del modelo. La segunda fase abarca desde el segundo punto de decisión hasta el final del horizonte de planificación, por lo tanto la segunda fase de la planificación táctica empieza cuando ya han transcurrido los periodos que van desde el inicio del ciclo de producción hasta el segundo momento de decisión (y ya se han ejecutado los planes de la primera fase entre ambos instantes). Los resultados de la primera fase en el segundo momento de decisión son datos de inicio para la segunda fase (inventario, pedidos pendientes, órdenes lanzadas, etc.). En la segunda fase se ejecuta otro modelo contra la demanda real (simulada); este modelo re-programa la producción desde el segundo momento de decisión hasta el fin del horizonte de planificación, respetando las capacidades contratadas de los proveedores lejanos y capacidades máximas de los proveedores locales.

10. Sobrecostes de las empresas de producción lejanas cuando se sobrepasa la capacidad contratada en un periodo dado: en la primera etapa del modelo no existe esta posibilidad pero en la segunda etapa, a partir del segundo momento de decisión, sí se puede decidir ejecutar más *strokes* de los que permite la capacidad contratada en la primera etapa, de manera que se tenga costes de sobre-utilización de la capacidad contratada al proveedor; así mismo, si no se utiliza toda la capacidad contratada se tiene el coste de sub-utilización (coste de oportunidad) por no aprovechar esa capacidad en un periodo dado (Chen y Lee, 2004).

En la tabla 3.9 se muestra un ejemplo de costes de sobre-utilización de la capacidad contratada (del proveedor lejano) y máxima (del proveedor local). En este ejemplo el segundo momento de decisión es en el periodo 4.

Penalidad +	Periodo					
Recurso	1	2	3	4	5	6
R1	0	0	0	5	5	5
R2	0	0	0	0.5	0.5	0.5
R3	0	0	0	6	6	6
R3e	0	0	0	7.5	7.5	7.5

Tabla 3.9: Costes de sobre-utilización de los recursos.

11. **Sobrecostes de las empresas de producción locales cuando se sobrepasa la capacidad máxima en un periodo dado:** en la primera etapa del modelo no existe esta posibilidad pero en la segunda etapa, a partir del segundo momento de decisión, sí se puede decidir ejecutar más *strokes* de los que permite la capacidad máxima (que es la capacidad instalada disponible local), de manera que se tenga costes de sobre-utilización de la capacidad máxima. Por lo contrario, existe coste de sub-utilización si no se utiliza toda la capacidad instalada disponible local (Peidro et al., 2007). La capacidad máxima es dato para el modelo, pero en cada ejecución se puede probar un valor distinto de la capacidad máxima, de esta manera se puede llegar a determinar cuánta capacidad instalada local conviene tener para determinadas características de la demanda, de los costes, etc.

    En la tabla 3.10 se muestra un ejemplo de costes de sub-utilización de la capacidad contratada (del proveedor lejano) y máxima (del proveedor local). En este ejemplo el segundo momento de decisión es en el periodo 4, y el recurso R2 no tiene coste de sub-utilización porque es un medio de transporte que se contrata sólo si es necesario y sólo en la cantidad requerida.

Penalidad -	Periodo					
Recurso	1	2	3	4	5	6
R1	0	0	0	1	1	1
R2	0	0	0	0	0	0
R3	0	0	0	1	1	1
R3e	0	0	0	0.1	0.1	0.1

Tabla 3.10: Costes de sobre-utilización de los recursos.

12. **Varios medios de transportes con costes, tiempos y capacidades diferentes:** el medio de transporte (avión, barco, etc.) es un recurso que tiene una capacidad máxima en cada periodo de tiempo. Cuando un *stroke* consume materiales en un país y entrega productos en otro país, incluye el transporte. Los costes, tiempos y consumo de la capacidad de los medios de transporte se deben incluir dentro de los *strokes*. Puede haber *strokes* que sólo realicen el transporte y cuenten con estos datos.

3.4. Datos del modelo propuesto

Los datos necesarios para el modelo matemático se muestran en las tablas 3.11 a, b y c.

Índices	**Descripción**
i, I	Cada ítem y el número total de ítems (los ítems son materias primas, componentes, subensambles y productos terminados) en diferentes ubicaciones.
t, T	Cada período y el número total de períodos (los periodos pueden ser semanas, quincenas o meses).
e, E	Cada escenario y el número total de escenarios (sólo para la demanda de productos terminados).
r, R	Cada recurso y el número total de recursos (los recursos son conjuntos de operarios y máquinas y/o transportes de los diferentes proveedores locales y extranjeros).
k, K	Cada *stroke* y el número total de *strokes* (los *strokes* son procesos que consumen ítems y recursos, y producen ítems de algún nivel superior en su lista de materiales, entregándolos en la misma o diferente ubicación).

Tabla 3.11 a: Índices.

Costes	**Descripción**
Precio de Venta (PV_i)	Precio unitario al que se vende cada producto terminado *i*. El PV_i es fijo durante toda la temporada de ventas.
Valor Residual (VR_i)	Precio de remate al que se vende cada del producto terminado *i* sobrante al final de la temporada de ventas.

	Si el número de unidades sobrantes es mayor que cierto porcentaje de la demanda total, el exceso se vende a una parte del valor residual (que puede ser 50%).
Coste de Pedidos Pendientes ($CB_{i,t}$)	Coste unitario de cada producto terminado *i* que falte para satisfacer la demanda en cada período *t*. Este es un coste de oportunidad que define la empresa en función de sus deseos de tener un mayor o menor nivel de servicio. Puede ser diferente en cada periodo *t*.
Coste de Inventario ($CH_{i,t}$)	Coste unitario de cada ítem *i* por periodo *t* en almacén. Es distinto para cada proveedor pero no necesita otro índice para identificar el proveedor porque el índice del ítem "*i*" lo identifica tanto como producto como en cuanto a su ubicación y proveedor.
Coste de Compra ($PC_{i,t}$)	Coste unitario de compra de las materias primas (ítem *i*) que compran los proveedores, en cada periodo *t* y es diferente para cada proveedor.
Costo de Operación ($CO_{k,t}$)	Coste unitario de ejecución de un *stroke k*. Puede ser diferente en cada periodo *t*. Es distinto para cada proveedor pero no necesita otro índice para identificar el proveedor porque el índice "*k*" lo identifica tanto como *stroke* como en cuanto a su proveedor.
Coste de la Capacidad ($CCAP_{r,t}$)	Coste unitario de la capacidad de cada recurso *r* de los proveedores. Puede ser diferente en cada periodo *t*.
Coste de Horas Extra ($PN^{+}_{r,t}$)	Coste unitario de uso de la capacidad de cada recurso *r* por encima de la capacidad contratada en caso de ser un proveedor lejano y por encima de la capacidad máxima en caso de ser un proveedor local, en cada periodo *t*.

Coste de Tiempo Ocioso ($PN^{-}_{r,t}$)	Coste unitario de la capacidad no usada de cada recurso *r* respecto de la capacidad contratada en caso de ser un proveedor lejano y de la capacidad máxima en caso de ser un proveedor local, por periodo *t*.

Tabla 3.11 b: Costes.

Parámetros	**Descripción**
Demanda ($D^{e}_{i,t}$)	Demanda de cada producto terminado *i* por periodo *t* para cada escenario *e* y sólo para el modelo de la fase uno.
Demanda ($D_{i,t}$)	Demanda de cada producto terminado *i* por periodo *t* y sólo para el modelo de la fase dos.
Probabilidad de cada escenario (P_e)	Probabilidad de ocurrencia de cada escenario *e* para el modelo de la fase uno.
Recursos por *stroke* ($RE_{r,k}$)	Capacidad de cada recurso *r* que consume cada *stroke k*. Igual para las dos fases del modelo.
Ítems que consume cada *stroke* ($M_{i,k}$)	Número de unidades de cada ítem *i* que consume cada *stroke k*. Igual para las dos fases del modelo.
Ítems que produce cada *stroke* ($N_{i,k}$)	Número de unidades del ítem *i* que produce el *stroke k*. El ítem que produce un *stroke k* es diferente de los ítems que consume dicho *stroke*.
Plazo de Entrega (LT_k)	Plazo de entrega (*lead tim*e) del *stroke k*. Este plazo abarca el tiempo de producción y el tiempo de transporte (si el *stroke* entrega el ítem generado en una ubicación diferente de donde consume materiales).

Capacidad máxima ($KAP_{r,t}$)	Capacidad máxima de cada recurso *r* de cada proveedor en cada periodo *t*.
Activa Compra ($AC_{i,t}$)	Indica si se puede comprar unidades de cada ítem *i* en cada periodo *t*. Sólo puede tomar valores 0 y 1.

Tabla 3.11 c: Parámetros.

3.5. Formulación del modelo matemático

En este apartado se presenta un modelo matemático que determina el plan de producción que maximiza el beneficio esperado de una cadena de suministro en la que se planifica contratar la capacidad de los recursos de los proveedores antes de conocer con certeza el comportamiento de la demanda, teniendo en cuenta recursos alternativos, ciclos de vida cortos, plazos de entrega largos, altos niveles de incertidumbre de la demanda, múltiples productos con lista de materiales conocida y componentes comunes, representando los procesos por medio del concepto de *stroke*.

El modelo matemático es un modelo de Programación Estocástica bi-Etapa, con dos momentos de decisión dentro del horizonte de planificación, en el que se asume que en un determinado instante, posterior al actual, la demanda se concretará, pero que en el momento de la primera decisión sólo se pueden estimar escenarios para la demanda (abarcando todo el horizonte de planificación). A dichos escenarios se les pueden estimar probabilidades, que se utilizan como la mejor aproximación a la incertidumbre de la demanda.

En el momento de la primera decisión, se determina cuánto contratar de capacidad de los recursos de los proveedores lejanos en función de los escenarios de demanda, abarcando todo el horizonte de planificación (ver la figura 3.9). El objetivo de este modelo es maximizar el beneficio ponderado de todos los escenarios, para todo el horizonte de ambas fases.

En el segundo momento de decisión, cuando se conoce con más certeza la demanda para lo que resta del horizonte de planificación, el objetivo es compensar los defectos de las decisiones de la primera etapa que por discrepancias entre la demanda

prevista y la demanda real en ambas fases, crean una diferencia entre lo decidido en la primera etapa y lo que se debe producir para la segunda etapa.

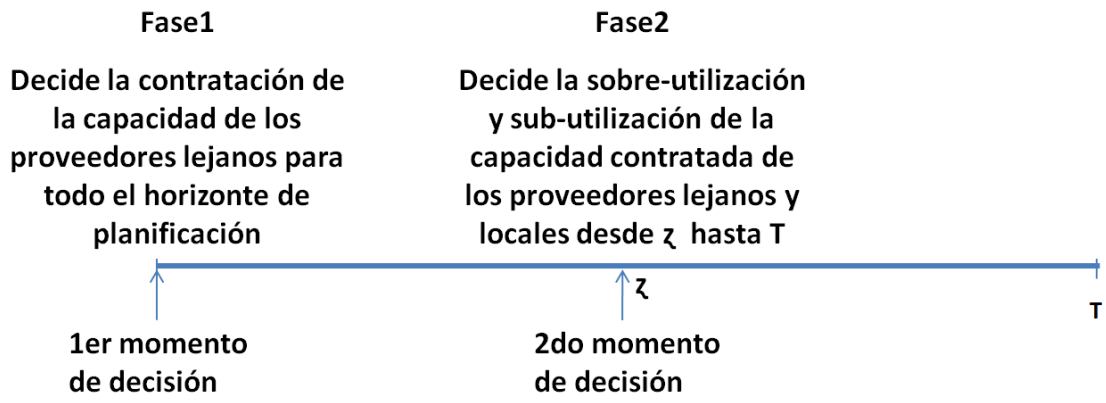


Figura 3.9: Fases del modelo matemático (elaboración propia)

Esta aproximación es similar a la utilizada por (Leung y Ng, 2007b) pero en nuestro modelo se permite el retraso en las entregas de los productos terminados. Además la introducción de listas de materiales complejas es sustancialmente más intuitiva gracias al concepto de *stroke* que permite modelar procesos alternativos con menos variables y restricciones, abarcando incluso el transporte entre proveedores y entre estos y el fabricante local.

La lógica general del modelo completo consiste en ejecutar de manera consecutiva las dos etapas. En el momento de la primera decisión se lanza el modelo de la primera etapa (Fase I), y luego en el momento de la segunda decisión se lanza el modelo de la segunda etapa (Fase II). Como el modelo completo se ejecuta en el momento de la primera decisión, se debe simular la demanda real para la Fase II. Como no se sabe si la demanda real será mayor o menor que la prevista en el primer momento de decisión, se debe simular muchas ocurrencias de la demanda real (tantas como el decisor estime convenientes) para obtener unos resultados del modelo completo que sean los más robustos ante las distintas demandas que pueden ocurrir en la segunda etapa.

3.5.1. Modelo Fase I

El modelo Fase I se formula como un problema de programación estocástica a través de un conjunto de escenarios asociados al comportamiento incierto

de la demanda, donde la ocurrencia de cada escenario tiene probabilidad P_e. El modelo Fase I abarca las decisiones tácticas de contratar la capacidad de los recursos en una cadena de suministro con proveedores locales y extranjeros, asimismo determina las cantidades a producir en cada proveedor, los inventarios, las entregas diferidas, etc.

Los índices, parámetros y variables del modelo se presentan en la tabla 3.12.

Índices			
I	Conjunto de los ítems -productos y componentes- ($i = 1 \ldots I$)		
T	Conjunto de los periodos durante el horizonte de planificación ($t = 1 \ldots T$)		
K	Conjunto de los strokes -operaciones y transportes- ($k = 1 \ldots K$)		
R	Conjunto de los recursos –plantas, medios de trasporte, etc.- ($r = 1 \ldots R$)		
E	Conjuntos de escenarios ($e = 1 \ldots E$)		
Parámetros			
PV_i	Precio de venta del ítem i.	$M_{i,k}$	Número de unidades de i que consume un stroke k.
VR_i	Valor residual del inventario del ítem i.	$N_{i,k}$	Número de unidades de i que genera un stroke k.
$CB_{i,t}$	Coste de faltante del ítem i en el periodo t.	LT_k	Lead time del stroke k.
$CH_{i,t}$	Coste de almacenamiento del ítem i en el periodo t.	$RE_{r,k}$	Consumo del stroke k del recurso r.
$PC_{i,t}$	Precio de compra del ítem i en el periodo t.	Q	Un número muy grande para que no restrinja.
$CO_{k,t}$	Coste de operación del stroke k en el periodo t.	$AC_{i,t}$	Número 0 ó 1 para autorizar compra del ítem i en el periodo t.
$CCAP_{r,}$	Coste unitario de contratar capacidad del recurso r en el periodo t	$KAP_{r,t}$	Capacidad máxima del recurso r en el periodo t
$D^e_{i,t}$	Demanda del ítem i en el periodo t.	P_e	Probabilidad de ocurrencia del escenario e
Variables			
$kap_{r,t}$	Capacidad a contratar del recurso r en el periodo t	$y^e_{i,t}$	Unidades almacenadas del ítem i en el periodo t en el escenario e.
$z^e_{k,t}$	Número de Strokes de tipo k a ejecutar en el periodo t en el escenario e.	$\beta^e_{i,t}$	Unidades faltantes del ítem i en el periodo t en el escenario e.
$v^e_{i,t}$	Ventas del ítem i en el periodo t en el escenario e.	$w^e_{i,t}$	Compras del ítem i en el periodo t en el escenario e.

Tabla 3.12: Índices, parámetros y variables del modelo estocástico.

El beneficio de cada escenario se calcula sumando los ingresos y restando los costes (ver la ecuación 3.1).

Los ingresos se obtienen por las ventas en cada periodo más el valor residual del inventario al final de productos terminados del horizonte de planificación. Los costes ligados a los productos son los relativos a los retrasos en las entregas (pedidos pendientes sólo de productos terminados), los costes de almacenamiento y las compras de material. Los costes ligados a las operaciones son los costes de los *strokes* (en los proveedores locales) y de la capacidad contratada (en los proveedores lejanos).

El objetivo del modelo, mostrado en la ecuación (3.1), es maximizar el beneficio esperado asociado a los escenarios menos los costes de la capacidad contratada. El coste de contratación de capacidad tiene el efecto más importante sobre el beneficio deseado (Li y Liu, 2008), por ello es importante su planificación.

$$Max(Z) = \sum_{e} P_e \left[\sum_{t}^{T} \sum_{i}^{I} \left(PV_i v_{i,t}^e + VR_i y_{i,T}^e - CB_{i,t} \beta_{i,t}^e - CH_{i,t} y_{i,t}^e - PC_{i,t} w_{i,t}^e \right) - \sum_{t}^{T} \sum_{k}^{K} CO_{k,t} z_{k,t}^e \right] - \sum_{t}^{T} \sum_{r}^{R} \left(CCAP_{r,t} kap_{r,t} \right) \qquad (3.1)$$

Las restricciones que aplican en este modelo se expresan a continuación.

El conjunto de restricciones (3.2) son las restricciones de continuidad del inventario, donde el inventario al final de un periodo se calcula a partir del inventario final del período anterior menos las salidas (ventas y consumos de los *strokes* respectivos) más las entradas (compras y productos de *strokes* realizados en el periodo que corresponde a su lead time).

$$y_{i,t}^e = y_{i,t-1}^e - v_{i,t}^e - \sum_{k=1}^{K} M_{i,k} z_{k,t}^e + w_{i,t}^e + \sum_{k=1}^{K} N_{i,k}\, z_{k,t-LT(k)}^e \qquad \forall_i \forall_t \forall_e \qquad (3.2)$$

Se admite que los productos se pueden comprar directamente a terceros para que el modelo sea lo más general posible, pero con las restricciones (3.3), se limitan tanto los productos a comprar como los periodos de compra.

Sólo se autoriza la compra de determinados productos. El parámetro AC indica si se puede comprar el material i en el periodo t. Si AC es 0, no se puede comprar. Si AC es 1, se puede comprar lo que se necesite pues Q es un valor muy grande.

$$w_{i,t}^{e} \leq AC_{i,t}Q \qquad \forall_i \forall_t \forall_e \qquad (3.3)$$

El conjunto de restricciones (3.4) son las restricciones de pedidos pendientes. Los pedidos pendientes de un periodo son los pedidos pendientes del periodo anterior menos las ventas más la demanda de ese periodo. Sólo los productos que tienen demanda tendrán ventas, y sólo éstos pueden tener pedidos pendientes, de esta manera el modelo se puede aplicar a un caso en el que los componentes puedan venderse.

$$\beta_{i,t}^{e} = \beta_{i,t-1}^{e} - v_{i,t}^{e} + D_{i,t}^{e} \qquad \forall_i \forall_t \forall_e \qquad (3.4)$$

Para garantizar un único resultado para los valores de los *strokes*, a pesar de tener varios escenarios, se incorporan las restricciones (3.5), que corresponden a decisiones que quedarán congeladas hasta el segundo momento de decisión (periodo τ), cuando se tenga más información que permita conocer la demanda con un grado mayor de certidumbre.

$$z_{k,t}^{e} = z_{k,t} \qquad \forall_k \forall_{t<\tau} \forall_e \qquad (3.5)$$

El modelo calcula la capacidad necesaria de cada recurso en cada periodo. Esto se expresa mediante el conjunto de restricciones (3.6) que suman los recursos utilizados por los *strokes* planificados.

$$\sum_k RE_{r,k} z_{k,t} \leq kap_{r,t} \qquad \forall_r \forall_t \qquad (3.6)$$

Se incorporan las restricciones (3.7) para garantizar que no se contrate más de la capacidad máxima disponible de cada recurso en cada proveedor. La variable $kap_{r,t}$ fija la capacidad contratada de los proveedores y no es modificable posteriormente. En la segunda etapa se puede sobre-utilizar o sub-utilizar esta capacidad contratada, pagando los respectivos costes de horas extra o de tiempo ocioso, pero no se puede cambiar las capacidades contratadas.

$$kap_{r,t} \leq KAP_{r,t} \qquad \forall_r \forall_t \qquad (3.7)$$

Restricciones de no negatividad:

$$kap_{r,t} \geq 0 \quad \forall_r \forall_t \,, \quad z_{k,t}^{e} \geq 0 \quad \forall_k \forall_t \,, \quad v_{i,t}^{e} \,, y_{i,t}^{e} \,, \beta_{i,t}^{e} \,, w_{i,t}^{e} \geq 0 \quad \forall_i \forall_t \forall_e \quad (3.8)$$

Con este modelo se determina en la primera fase cuánta capacidad se debe contratar para maximizar el beneficio esperado antes de tener certeza del comportamiento de la demanda. Además se determinan las cantidades a producir de cada ítem en cada proveedor así como los almacenamientos y transportes. La capacidad que se decida contratar en cada uno de los recursos de los proveedores lejanos se mantendrá fija en la segunda etapa del modelo, mientras que las cantidades a producir de cada ítem, los almacenamientos y los transportes sólo permanecerán fijos hasta el segundo momento de decisión (en el que se ejecuta el modelo Fase II).

3.5.2. Modelo Fase II

El modelo diseñado para la fase II tiene como entrada la capacidad que fue contratada a los proveedores lejanos previamente en la fase I, los inventarios de todos los ítems en el segundo momento de decisión (τ), los pedidos pendientes de productos terminados en el segundo momento de decisión y las recepciones programadas de componentes, subensambles y pt. El objetivo de este modelo es maximizar el beneficio de lo que resta del horizonte de planificación ($t \geq \tau$) a través de minimizar la discrepancia entre lo planificado en la primera etapa y lo que conviene producir para maximizar los beneficios a partir del segundo momento de decisión. La discrepancia en el uso de la capacidad contratada estará representada por las variables $F_{r,t}^{+}$ y $F_{r,t}^{-}$.

$F_{r,t}^{+}$ es la capacidad extra que requiere ser contratada del recurso *r* en el periodo $t \geq \tau$, y $F_{r,t}^{-}$ es la capacidad no utilizada del recurso *r* en el periodo $t \geq \tau$.

Para calcular los ingresos de rematar los ítems sobrantes se agregan las variables R_{1i} y R_{2i} . La primera es la cantidad de cada producto final que queda en inventario al final del ciclo de ventas (periodo *t* = T), siempre y cuando esta cantidad sea menor o igual que el 10 por ciento de la demanda total del respectivo ítem. En cambio R_{2i} es la cantidad que queda en inventario por encima del 10 por ciento de la demanda total del respectivo producto final. Ese porcentaje (10 por ciento) puede ser diferente para cada producto terminado. La primera cantidad, R_{1i}, será rematada a su valor residual, mientras la segunda cantidad, R_{2i}, será rematada al 50% de su valor residual (este nuevo porcentaje también puede ser diferente para cada producto).

Para la ejecución del modelo de la fase II se requiere de parámetros y variables adicionales a los de la fase I. Los parámetros adicionales son:

- La capacidad $CAP_{r,t}$ contratada en la fase I del modelo, del recurso *r* de cada proveedor lejano para el periodo *t*,
- las recepciones planificadas $RPL_{i,t}$, que corresponden a *strokes* lanzados en periodos anteriores al segundo momento de decisión y cuyo plazo de entrega aún no se ha cumplido en el periodo del segundo momento de decisión,
- la demanda total DT_i de cada producto terminado (sumatoria de su demanda desde $t = 0$ hasta $t = T$),
- por último, los costes de penalidad $PN^{+}_{r,t}$, por la capacidad extra que se requiere para satisfacer la demanda y por la capacidad contratada no utilizada $PN^{-}_{r,t}$.

Las variables adicionales son: R_{1i} , R_{2i} , $F^{+}_{r,t}$ y $F^{-}_{r,t}$.

La capacidad máxima de los proveedores locales se mantiene constante en la segunda etapa (desde el segundo momento de decisión hasta el fin del horizonte de planificación), es decir se mantiene igual que en la primera etapa, y $CAP_{r,t}$ asume dicho valor sólo para los proveedores locales. Para los proveedores lejanos, $CAP_{r,t}$ es la capacidad contratada determinada en el modelo de la fase I para $t \geq \tau$.

El modelo se presenta en las ecuaciones (3.9) a la (3.16).

La función objetivo es igual que en la fase I salvo por tres motivos:

- porque no tiene escenarios ya que en esta segunda etapa la demanda es conocida,
- porque suma los ingresos por remate de los productos sobrantes (una parte del inventario final de productos terminados (R_{1i}) por el valor residual (VR_i) y el resto del inventario final (R_{2i}), si hubiera, por la mitad del valor residual),

- y porque no se resta el coste de la capacidad sino el coste de sobre-utilización y sub-utilización de la capacidad contratada (en los proveedores lejanos) y de la capacidad máxima (en los proveedores locales).

$$Max(Z) = \sum_{t=\tau}^{T}\left[\sum_{i}^{I}\left(PV_i v_{i,t} - CB_{i,t}\beta_{i,t} - CH_{i,t}y_{i,t} - PC_{i,t}w_{i,t}\right) - \sum_{k}^{K} CO_{k,t}z_{k,t}\right] + VR_i(R_{1i} + 0.5R_{2i}) - \sum_{t=\tau}^{T}\sum_{r}^{R}\left(PN_{r,t}^{+}F_{r,t}^{+} + PN_{r,t}^{-}F_{r,t}^{-}\right) \qquad (3.9)$$

Las restricciones también son iguales a las de la fase I excepto las restricciones 3.5, 3.6 y 3.7, que no se emplean en la fase II. En cambio la restricción de inventario (ecuaciones 3.10) tiene un término adicional que es $RPL_{i,t}$, las recepciones planificadas.

$$y_{i,t} = y_{i,t-1} - v_{i,t} + RPL_{i,t} - \sum_{k=1}^{K} M_{i,k}z_{k,t} + w_{i,t} + \sum_{k=1}^{K} N_{i,k}z_{k,t-LT(k)} \qquad \forall_i\forall_t \quad (3.10)$$

$$w_{i,t} \le AC_{i,t}Q \qquad \forall_i\forall_t \quad (3.11)$$

$$\beta_{i,t} = \beta_{i,t-1} - v_{i,t} + D_{i,t} \qquad \forall_i\forall_t \quad (3.12)$$

Además se agrega la restricción de capacidad (el conjunto de ecuaciones 3.13) para calcular la cantidad de capacidad extra necesaria o de capacidad sobrante para cada recurso en cada periodo. En este conjunto de ecuaciones, $CAP_{r,t}$ asume el valor de la variable de capacidad $kap_{r,t}$ del modelo Fase I para los proveedores lejanos y asume el valor de la capacidad máxima (dato) de los proveedores locales.

$$\sum_{k} RE_{r,k}z_{k,t} + F_{r,t}^{-} - F_{r,t}^{+} = CAP_{r,t} \qquad \forall_r\forall_t \quad (3.13)$$

Con la restricción (3.13) se busca la utilización exacta de la capacidad contratada en el primer momento de decisión, puesto que cuando en algunos periodos se subutiliza o se requiere más capacidad se aumenta los costes por la penalización. Con los valores obtenidos en las variables asociadas a la capacidad se debe negociar con los proveedores. Según Li y Liu (2008), cuando hay coordinación con los proveedores y se ha reservado la capacidad con antelación, el sistema funciona de mejor manera que cuando no se coordina.

Por último, se agregan las restricciones (3.14) y (3.15) sobre las cantidades sobrantes de cada producto terminado. Las ecuaciones (3.14) reparten el inventario final *(t=T)* de cada producto terminado entre R_{1i} y R_{2i}, y las ecuaciones (3.15) limitan el valor de R_{1i} al 10 por ciento de la demanda total del respectivo ítem. Este porcentaje puede ser variado cada vez que se ejecute el modelo.

$$y_{i,T} = R_{1i} + R_{2i} \qquad \forall_i \qquad (3.14)$$

$$R_{1i} \leq 0.1DT_i \qquad \forall_i \qquad (3.15)$$

Restricciones de no negatividad:

$$F^{+}_{r,t}, F^{-}_{r,t} \geq 0 \quad \forall_r \forall_t, \; z_{k,t} \geq 0 \quad \forall_k \forall_t, \; v_{i,t}, y_{i,t}, \beta_{i,t}, w_{i,t} \geq 0 \quad \forall_i \forall_t \; (3.16)$$

3.5.3. Comparación del modelado con *strokes* con otros enfoques

En este apartado se pretende ilustrar la ventaja que de usar el concepto de *stroke* en la planificación de las operaciones de CS y para ello se propone la comparación con otros enfoques, con un caso sencillo.

El caso es el de una CS en la que se debe planificar la producción de artículos de innovación con componentes comunes y no comunes, y con dos alternativas de aprovisionamiento, un fabricante local, caro y con plazo de entrega corto, y un proveedor lejano, barato y con plazo de entrega largo. Los dos "proveedores" (el proveedor lejano y el fabricante local) tienen diferentes recursos (conjuntos de máquinas y/u operarios) para los cuales es conocida su capacidad máxima de producción en horas normales y capacidad máxima de horas extra. Se conoce además su ubicación, todos sus costes de producción e inventario, asimismo los costes de horas extra y de tiempo ocioso, sus tiempos de producción y, los tiempos y costes de transporte entre ellos.

Supóngase que las CS vende dos productos A y E a un solo mercado ubicado en el país del fabricante local. La demanda es prevista y se ingresa como dato para todo el horizonte de planificación. El proceso de fabricación tiene dos etapas: transformación de materias primas en componentes y ensamble de los mismos. La compra de materias primas y la transformación de estas en componentes se realizan en el proveedor lejano (por economía de costes) y éste también puede hacer el ensamble

final. El ensamble final también puede ser hecho por el fabricante local. Además se puede fabricar cada producto terminado de un solo tirón desde materias primas hasta producto terminado (producción directa) y se puede aplazar las etapas de proceso (manteniendo inventarios intermedios) tanto del proveedor lejano como en el fabricante local (producción indirecta).

Existe un único producto intermedio (componente C) que puede ser almacenado por el proveedor lejano (antes de su envío o transporte) y/o por el fabricante local (después del transporte). Los productos A y E se pueden obtener desde el mismo componente C a través de procesos que deben ser realizados por el proveedor lejano o por el fabricante local.

Existe una única materia prima D que se puede transformar en componente C o directamente en producto terminado A o E.

Modelado con el enfoque de *strokes*

Con el enfoque de modelado con *strokes*, como se ha mencionado, el índice de un producto no sólo lo distingue de otros ítems sino también informa la ubicación en que se encuentra. Así el único producto intermedio puede recibir dos nombres: B y C dependiendo del lugar dónde esté ubicado: B está ubicado en el país de destino (país del fabricante local) y C en el país de origen (país del proveedor lejano).

Con el concepto de *strokes* se pueden proponer ocho formas de obtener los productos A y E, de manera que se pueda aplazar el transporte y el ensamble final. Ver la figura 3.10. en la siguiente página.

El modo más “directo” (es decir sin inventarios intermedios) es que la compra de la materia prima y su transformación hasta generar el producto A y su desplazamiento hasta el país de destino sean una única operación hecha por el *stroke Sk1*. El *Sk1* tendría su coste y su *lead time* que incluirían los costes y tiempos de producción y transporte respectivamente.

Una segunda alternativa es que la transformación se haga en el país de origen y se transporte inmediatamente hasta el proveedor local con el *Sk2*, donde

queda almacenada como componente B, a la espera de ser transformado en producto terminado A mediante el *Sk3*. El almacenamiento no es parte de ningún *stroke*.

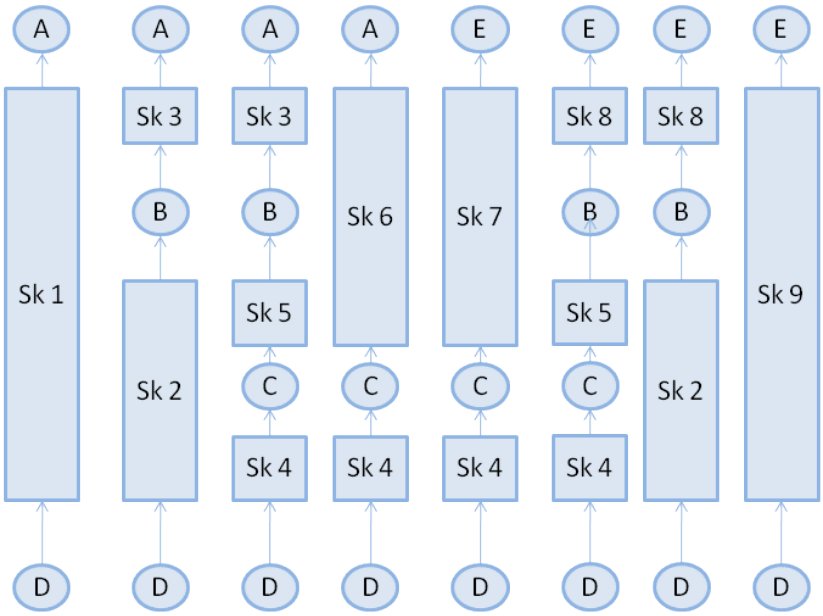


Figura 3.10: Alternativas en las operaciones de la CS.

Una tercera opción sería que una vez realizada la transformación de la materia prima a producto intermedio con el *Sk4*, el prodtucto quedara almacenado en el país de origen como componente C, desde allí se transportaría con el *Sk5* al país de destino, donde se almacenaría nuevamente como componente B, para luego mediante el *Sk3* convertirse en producto A.

La cuarta alternativa implicaría una transformación *Sk4* y almacenamiento en país de origen como componente C, para posteriormente ser transformado en producto terminado en el mismo país de origen y luego transportado al país de destino en una operación *Sk6*. Como se puede apreciar, las alternativas segunda, tercera y cuarta son formas de "producción indirecta" del producto A.

La quinta alternativa es igual que la cuarta salvo que se emplea el *Sk7* en vez del *Sk6* para producir el producto terminado E.

Las alternativas sexta y séptima son iguales que la tercera y segunda respectivamente, salvo porque se emplea el *Sk8* en lugar del *Sk3* para producir E.

La octava alternativa es igual que la primera salvo porque se emplea el *Sk9* en vez del *Sk1* para producir E sin almacenamientos intermedios.

Este, aparentemente complejo sistema, refleja bastante bien el juego de alternativas que tiene que plantear un importador de productos de plástico donde la inyección y el ensamble serían las dos operaciones básicas de dos productos innovadores que enfrentan una demanda estacional y volátil (cuya previsión es más imprecisa a medida que la anticipación respecto de la temporada de ventas es mayor).

Todo este complejo sistema queda reducido a planificar la ejecución de un juego muy reducido de operaciones (sólo nueve *strokes*) como se representa en la figura 3.11.

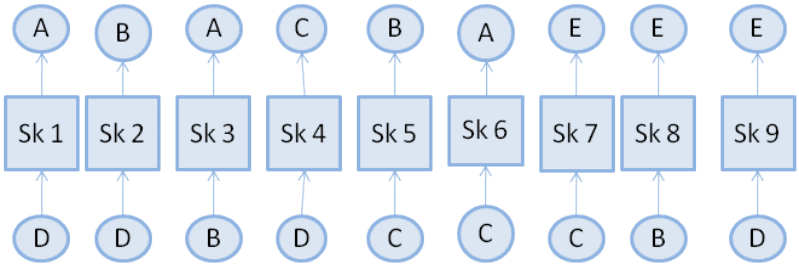


Figura 3.11: Los *strokes* del caso propuesto.

Con la estructura de *strokes* propuesta tendríamos 4 procesos para transformar la materia prima D en A, B, C y E, que son los *strokes Sk1*, *Sk2*, *Sk4* y *Sk9* respectivamente. El componente B se puede transformar en A o E según se procese en los *strokes Sk3* o *Sk8* respectivamente. Y el componente C se puede tranformar en B, A o E según se procese en los *strokes Sk5*, *Sk6* o *Sk7* respectivamente. De esta manera se analizarían todas las alternativas de producción directa e indirecta de los productos finales.

En el caso propuesto se comprarían externamente únicamente los productos D, pero la lógica del modelo permitiría comprar B y C.

Comparación con otros enfoques

Con el enfoque tradicional (sin *strokes*) se planifica la producción de ítems (en vez de la ejecución de los *strokes*). Además, con el enfoque sin *strokes* se clasifican los ítems en materias primas, componentes, subensambles y productos finales asignándolos a distintos sets o conjuntos (Escudero et al. (1999) y Alonso-Ayuso et al. (2003)), aumentando de esta manera el número de términos en las restricciones y el número de restricciones necesarias respecto del enfoque con *strokes*.

En enfoques más recientes, como el de Peidro et al. (2007) y el de Calderón et al. (2008) (ver el modelo completo en el Anexo 1), se emplea un solo set con el índice ("*i*") para todos los ítems excepto para diferenciar los productos directos de los indirectos. Leung y Ng (2007 a y b) emplea diferentes sets de índices para distinguir los productos finales hechos a partir de materias primas de aquellos hechos a partir de subensambles (así se tienen que poner tantos índices como distintas alternativas haya de producción indirecta de un producto final), y en Calderón et al. (2008) se agrega un set con los índices "*ip*" para los productos que consumen otros ítems para su elaboración (sólo las materias primas no son *ip*) de manera que se pueda usar la lista de materiales (ver el último término del conjunto de restricciones de inventario (3.17)). Las variables y los índices están definidos en el Anexo 1.

$$Inv_{\text{int}} = Inv_{in,t-1} + \sum_{j}^{J} P_{injt} + \sum_{no}^{NO}\sum_{l}^{L} RTN_{ino,nd=n,lt} + CC_{\text{int}} - \\ - \sum_{nd}^{ND}\sum_{l}^{L} TNN_{i,no=n,ndlt} - S_{\text{int}} - \sum_{ip=1}^{IP}(LMI_{ip,in}\sum_{j}^{J} P_{i=ip,njt}) \qquad \forall\ i, n, t \tag{3.17}$$

En el enfoque con *strokes* se emplea sólo un set con los índices ("*i*") para todos los ítems, y no es necesario distinguir los productos terminados en directos o indirectos (y sus distintas alternativas de producción indirecta), pues de esto se encargan los *strokes* (ver el producto A y los *strokes Sk1, Sk3* y *Sk6* en la figura 3.11). Tampoco es necesario usar el índice *ip* pues los *strokes* consumen los ítems (información que se ingresa al modelo matemático mediante la lista de materiales $M_{i,k}$ (ver las restricciones de inventario (3.18)).

$$y_{i,t}^{e} = y_{i,t-1}^{e} - v_{i,t}^{e} - \sum_{k=1}^{K} M_{i,k} z_{k,t}^{e} + w_{i,t}^{e} + \sum_{k=1}^{K} N_{i,k}\, z_{k,t-LT(k)}^{e} \qquad \forall_i \forall_t \forall_e \tag{3.18}$$

Además, usar índices diferentes para los productos directos e indirectos aumenta términos en las restricciones de capacidad para cada recurso en cada proveedor y periodo (ver las ecuaciones (3,19) de Calderón et al., (2008)). Sin esta distinción, con el enfoque de *strokes*, sólo se suman las capacidades usadas por los *strokes* en cada recurso y periodo (ver las ecuaciones (3,20)).

$$\sum_{i=1}^{I} TP_{inj}\left(\sum_{id}^{ID} P_{i=id,nJt} LMD_{id,in}\right) + \sum_{i=1}^{I} P_{injt} TP_{inj} \leq MAXPN_{njt} + MAXHE_{njt}$$

$$\forall\ n, j, t \qquad (3,19)$$

$$\sum_{k} RE_{r,k} z_{k,t} \leq kap_{r,t} \qquad \forall_r \forall_t \qquad (3,20)$$

Se puede notar que la restricción de capacidad con *strokes* tampoco necesita el índice *n* (set de nodos que identifican a los proveedores) pues cada *stroke* está identificado como operación ligada a los recursos "*r*" de un o unos proveedores específicos.

También las restricciones de horas extra y tiempo ocioso se simplifican. Se pueden ver las ecuaciones (3.21) de Calderón et al. (2008) y las (3.22) con *strokes* y comprobar que las ecuaciones (3.22) no tienen *id* ni *n*.

$$Tex_{njt} = \sum_{i=1}^{I} TP_{inj}\left(\sum_{id}^{ID} P_{i=id,nJt} LMD_{id,in}\right) + \sum_{i=1}^{I} P_{injt} TP_{inj} - MAXPN_{njt} + Toc_{njt} \quad \forall\ n, j, t \quad (3.21)$$

$$\sum_{k} RE_{r,k} z_{k,t} + F_{r,t}^{-} - F_{r,t}^{+} = CAP_{r,t} \qquad \forall_r \forall_t \qquad (3.22)$$

En la restricción de entregas diferidas de Calderón et al. (2008) se tiene que emplear un término adicional para los productos directos (ver las ecuaciones (3.23)), mientras que eso no es necesario en la restricción del modelo con *strokes* (ver las ecuaciones (3.24)). Además ésta última no requiere el índice *n*. En general, todas las restricciones del modelo con *strokes* no requieren el índice *n*.

$$ED_{\text{int}} = ED_{in,t-1} + D_{\text{int}} - S_{\text{int}} - S_{i+F,nt} \qquad \text{para } i=1,\ldots,F,\ \forall\ n, t \qquad (3.23)$$

$$\beta_{i,t} = \beta_{i,t-1} - v_{i,t} + D_{i,t} \qquad \forall_i \forall_t \qquad (3.24)$$

Dado que se requiere hacer envíos entre nodos (transporte de ítems entre proveedores y fabricante local), los enfoques tradicionales (Escudero et al. (1999), Alonso-Ayuso et al. (2003), Peidro et al. (2007) y Calderón et al. (2008)) consideran variables de decisión para los envíos y las recepciones en las restricciones de inventario (ver variables *RTN* y *TNN* en las ecuaciones (3.17)) y en las restricciones que equilibran los envíos con las recepciones (ver las ecuaciones (3.25) correspondientes a Peidro et al. (2007)).

$$RTN_{inondlt} = RTNI_{inondlt} + TNN_{inondlt-TTN_{nondl}} \qquad \forall\ i, no, nd, l, t \qquad (3.25)$$

Además los enfoques tradicionales han tenido que definir tres sets adicionales de índices: el de nodos de origen (*no*), el de nodos de destino (*nd*) y el de medios de transporte (*l*).

Con el enfoque de *strokes*, el *stroke* consume y produce items que contienen la información de su ubicación (por ejemplo el *Sk1* consume D del país de origen y entrega A en el país de destino), por esto no se necesitan los sets de nodos de origen y destino; así mismo algunos *strokes* incluyen el transporte (los *strokes Sk1, Sk2, Sk5, Sk6, Sk7* y *Sk9*) y por ello incluyen la información del medio que usan y no es necesario el set de medios de transporte. Por lo anterior, en la ecuación de inventarios con *strokes* no aparecen ni las variables de envío y recepción ni los índices de nodos de origen, nodos de destino y medios de transporte (ver las ecuaciones 3.18). Así mismo el modelo con *strokes* no requiere las ecuaciones de equilibrio de los envíos con las recepciones (3.25).

En los enfoques tradicionales, en la función objetivo se deben considerar los costes de transporte (ver *CTN* en las ecuaciones (3.26) de Calderón et al. (2008)). En el enfoque con *strokes* no se necesario pues cada *stroke* que incluye transporte incluye el coste del transporte (ver las ecuaciones (3.27)).

$$Max(Z) = \sum_{i}^{I}\sum_{n}^{N}\sum_{t}^{T} PV_{\text{int}} S_{\text{int}} - \sum_{i}^{I}\sum_{n}^{N}\sum_{j}^{J}\sum_{t}^{T} CP_{injt} P_{injt} - \sum_{n}^{N}\sum_{j}^{J}\sum_{t}^{T} (CPE_{njt} Tex_{njt} + CPO_{njt} Toc_{njt})$$
$$- \sum_{i}^{I}\sum_{n}^{N}\sum_{t}^{T} (CMP_{\text{int}} CC_{\text{int}} + CIN_{\text{int}} Inv_{\text{int}} + CED_{\text{int}} ED_{\text{int}}) - \sum_{i}^{I}\sum_{no}^{NO}\sum_{nd}^{ND}\sum_{l}^{L}\sum_{t}^{T} CTN_{inondlt} TNN_{inondlt} \qquad (3.26)$$

$$Max(Z) = \sum_{e} P_e \left[\sum_{t}^{T}\sum_{i}^{I} \left(PV_i v_{i,t}^e + VR_i y_{i,T}^e - CB_{i,t}\beta_{i,t}^e - CH_{i,t} y_{i,t}^e - PC_{i,t} w_{i,t}^e \right) - \sum_{t}^{T}\sum_{k}^{K} CO_{k,t} z_{k,t}^e \right] - \sum_{t}^{T}\sum_{r}^{R} \left(CCAP_{r,t} kap_{r,t} \right) \qquad (3.27)$$

Otra importante ventaja del modelo con *strokes* frente a los modelos existentes (publicados hasta la fecha) es que el modelo con *strokes* además de determinar la capacidad que conviene contratar de cada proveedor en cada periodo de

tiempo, permite pagar por capacidad y pagar por producción. Este modelo permite pagar por la capacidad de los recursos contratados a proveedores lejanos y permite pagar por producto fabricado a los proveedores locales. En ambos casos también permite pagar horas extra y tiempo ocioso, dependiendo del tipo de contrato con cada proveedor.

En el modelo Fase I propuesto en esta tesis, se incluyen los costes de las capacidades contratadas (ver el último término de la función objetivo (3.27)), que no necesariamente coinciden con las capacidades máximas de los proveedores lejanos, y se incluyen los costes de los *strokes* ejecutados en los proveedores locales (ver el penúltimo término de la (3.27)).

En el modelo Fase II se incluyen los costes de horas extra y tiempo ocioso (ver las ecuaciones (3.28)). Como los modelos Fase I y Fase II se ejecutan consecutivamente, se consideran todos los costes mencionados.

$$Max(Z) = \sum_{t=\tau}^{T}\left[\sum_{i}^{I}\left(PV_i v_{i,t} - CB_{i,t}\beta_{i,t} - CH_{i,t}y_{i,t} - PC_{i,t}w_{i,t}\right) - \sum_{k}^{K} CO_{k,t}z_{k,t}\right] + VR_i(R_{1i} + 0.5R_{2i}) - \sum_{t=\tau}^{T}\sum_{r}^{R}\left(PN_{r,t}^{+}F_{r,t}^{+} + PN_{r,t}^{-}F_{r,t}^{-}\right) \qquad (3.28)$$

3.6. Implementación de los modelos propuestos

Para la implementación de los dos modelos (fase I y fase II), se ha desarrollado una herramienta en Java que toma los datos (ingresados en tablas de Excel) y los modelos matemáticos (programados en lenguaje AMPL) y los hace resolver por el optimizador comercial Gurobi Optimizer 3.0.3. o el GNU LpSolve (está preparada para usar uno u otro optimizador), luego promedia los resultados (según el número de ocurrencias) y los vuelve a presentar en hojas de Excel. En el Anexo 2 se muestran los algoritmos hechos en Java que gestionan la información desde los datos iniciales, calculando los valores que pasan de la fase I a la fase II, hasta la generación del archivo de resultados.

El modelo total incluye, además de los modelos Fase I y Fase II, cinco algoritmos que realizan las siguientes funciones:

1. Lectura e ingreso de datos al modelo Fase I.

2. Generación de las ocurrencias de demanda con el método de simulación de Monte Carlo y el método de difusión de Bass.

3. Lectura de los resultados del modelo Fase I e identificación de las capacidades contratadas de los proveedores lejanos para los periodos desde $t = \tau$ hasta $t = T$, y de las recepciones programadas pendientes en el periodo $t = \tau$. Además, cálculo de los inventarios y pedidos pendientes en el periodo $t = \tau - 1$, e ingreso de datos al modelo Fase II.

4. Ejecución del modelo Fase II tantas veces como el número de ocurrencias de la demanda.

5. Recolección de los resultados de las ocurrencias de la Fase II, cálculo de promedios, suma de costos con los costos de la Fase I, generación del archivo de resultados en XML y escribir los resultados en tablas de Excel.

El proceso se inicia al extraer de la base de datos la información de los parámetros que alimentan los modelos y que junto con los escenarios de demanda generan el archivo Datos.XML (*Extensible Markup Language*), el cual contiene toda la información de entrada al Módulo Solver para la ejecución de la primera fase (ver la Figura 3.12). Una vez ejecutado el modelo de la fase I, se pasa información de las capacidades contratadas $CAP_{r,t}$ (de los proveedores lejanos) para todo el horizonte de planificación y de las recepciones planificadas $RPL_{i,t}$ (pendientes en el momento de la segunda decisión) a la fase II, junto con el cálculo de los inventarios y pedidos pendientes en el instante de la segunda decisión dada la ocurrencia de la demanda real (archivo DatosII.XML). En $t = \tau$ (segundo momento de decisión) se lanza el modelo de la fase II y se genera un archivo Resultados.XML con los resultados obtenidos del modelo de la fase II. Estos resultados indican qué cantidades producir de cada producto y cómo usar las capacidades previamente contratadas con sus proveedores para minimizar los costes de sobre-utilización y sub-utilización de las capacidades y maximizar su expectativa de beneficios.

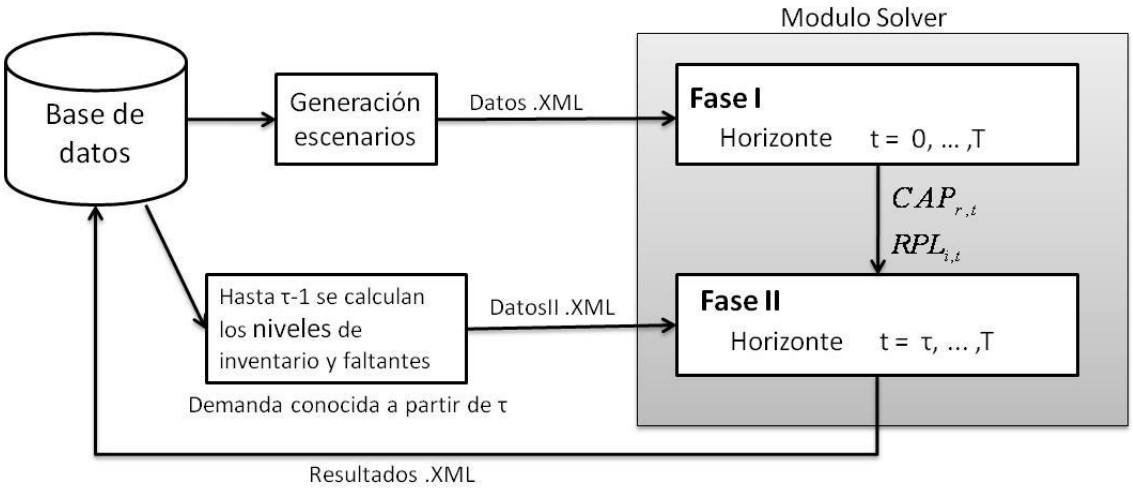


Figura 3.12: Lógica general para la resolución de los problemas

El modelo Fase II, que se ejecuta en $t = \tau$, utiliza simulaciones de la demanda con el objeto de comprobar la calidad del resultado obtenido en la primera fase. Se generan diferentes ocurrencias de demanda utilizando la simulación de Monte Carlo por medio de un modelo de difusión para determinar la distribución de la demanda por periodos (Chen et al., 2002). Se utilizan los modelos de difusión porque los modelos clásicos de previsión se basan en la extrapolación de los datos de las ventas históricas del producto, pero cuando no existen datos históricos las técnicas tradicionales de previsión no son aplicables (Morrison, 1995). Los modelos de difusión, ampliamente utilizados en marketing (Meade e Islam, 2006), tienen el propósito de estimar las ventas de los productos nuevos antes del lanzamiento real de los mismos (Scitovski y Meler, 2002) cuando no existe información histórica de los mismos sino de patrones de comportamiento de productos similares. El modelo de difusión utilizado en nuestro caso es el propuesto por Bass (1969), el cual asume que las adopciones de los consumidores no sólo dependen de la influencia interna generada por la presión social (de los pioneros que adoptan primero el producto) sino también por influencias externas, como lo es la publicidad. El modelo se plantea a través de la ecuación diferencial 3.29, en la que *N(t)* es el número acumulado de clientes que han adoptado el producto nuevo hasta el instante *t*, *p* es el coeficiente de innovación, *q* es el coeficiente de imitación y *m* es el potencial del mercado. Su solución (ecuación 3.30) muestra el número acumulado de clientes que han adoptado el producto nuevo hasta el instante *t*.

$$\frac{dN(t)}{dt} = \left(p + qN(t)\right)\left(m - N(t)\right) \tag{3.29}$$

$$N(t) = m\frac{1 - e^{-(p+q)t}}{1 + \frac{q}{p}e^{-(p+q)t}} \tag{3.30}$$

Para generar las demandas, en esta tesis, se usa la simulación de Monte Carlo asumiendo que los parámetros *p* y *q* siguen una distribución beta y el parámetro *m* una distribución triangular (Sohn y Lim, 2008). Con estas distribuciones, para cada ocurrencia se generan los parámetros y con estos de calcula el número de adoptantes no acumulado y acumulado para cada periodo. Un ejemplo de la codificación para generar las demandas con el modelo de difusión de Bass se muestra en la figura 3.13.

```
01 public GenerarDemanda{
02
03 public double numT; //Numero de periodos
04 public double numE; //Numero de escenarios
05
06 public double m = new double[numE]; //Mercado potencial
07 public double p = new double[numE]; //Tasa innovadores
08 public double q = new double[numE]; //Tasa Imitadores
09 public double N = new double [numT][numE]; //Adoptantes acumulados periodos
10 public double n = new double [numT][numE]; //Adoptantes por periodo
11
12   public void Generar(){
13
14   for(int e=0;e<numE;e++{
15
16       m[e]=GenerarM(M1,M2,M3); //Generando con Distr. Triangular
17       p[e]=GenerarP(alfa1,beta1); //Generando con Distr. Beta
18       q[e]=GenerarQ(alfa2,beta2); //Generando con Distr. Beta
19
20       for(int t=0;t<N;t++){
21
22         N[t][e]=CalcularN(t,m[e],p[e],q[e]);
23         n[t][e] Calcularn(t,m[e],p[e],q[e]);
24       }
25     }
26   }
27 }
```

Figura 3.13: Algoritmo para generar demandas utilizando el modelo de Bass

A continuación se muestra la generación siete demandas por medio de la simulación de Monte Carlo y el método de difusión de Bass; los parámetros siguen las distribuciones mostradas en la tabla 3.13. Utilizando el algoritmo descrito en la figura 3.13 se generan las demandas que se muestran en las figuras 3.14 y 3.15.

Parámetro	Distribución	Media	Varianza
m	Triangular (4000, 4350, 5200)	4433.33	
p	Beta (0.03, 0.09)	0,05	0,004
q	Beta (0.4, 0.99)	0,071	0,008

Tabla 3.13: Distribución para cada parámetro

En las figuras 3.14 y 3.15 el eje horizontal muestra el tiempo en días y en el eje vertical la demanda.

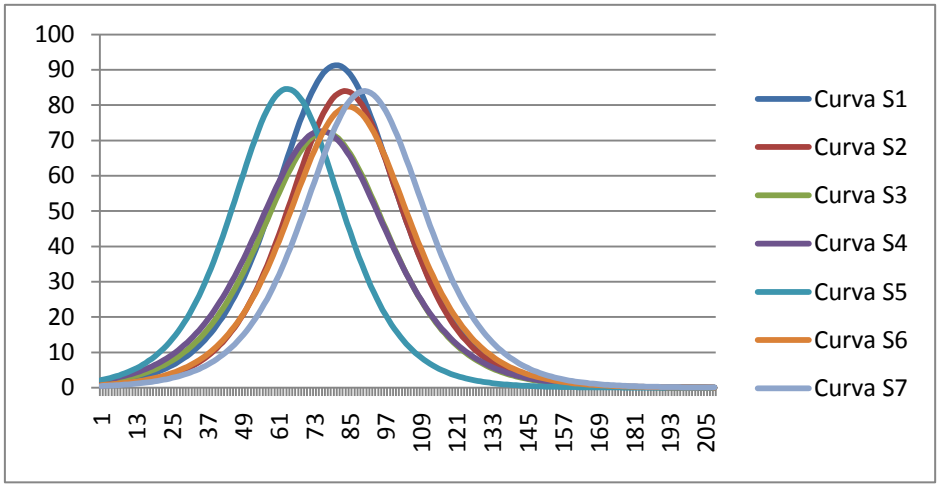


Figura 3.14: Número de adoptantes para el periodo t

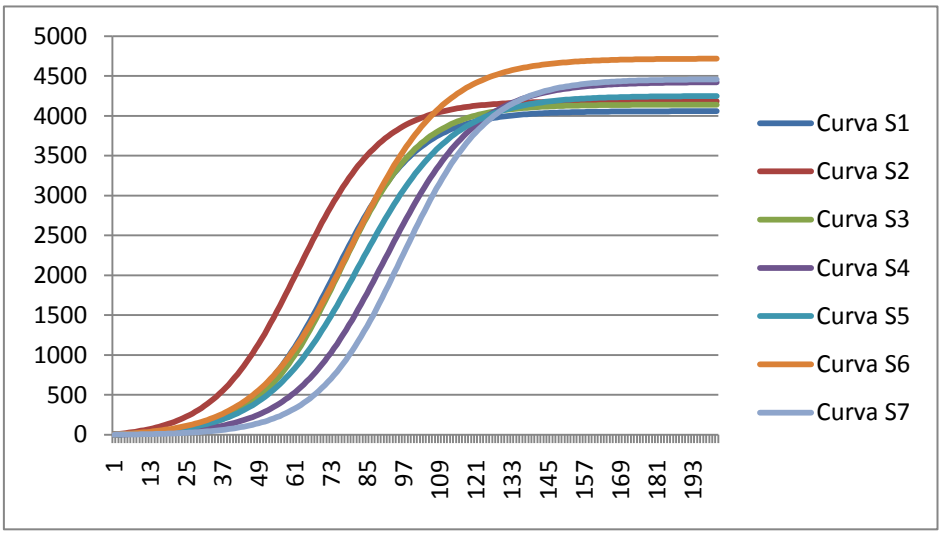


Figura 3.15: Número acumulado de adoptantes para el periodo t

La lógica de la implementación con simulación de la demanda para la segunda fase, se muestra en la figura 3.16.

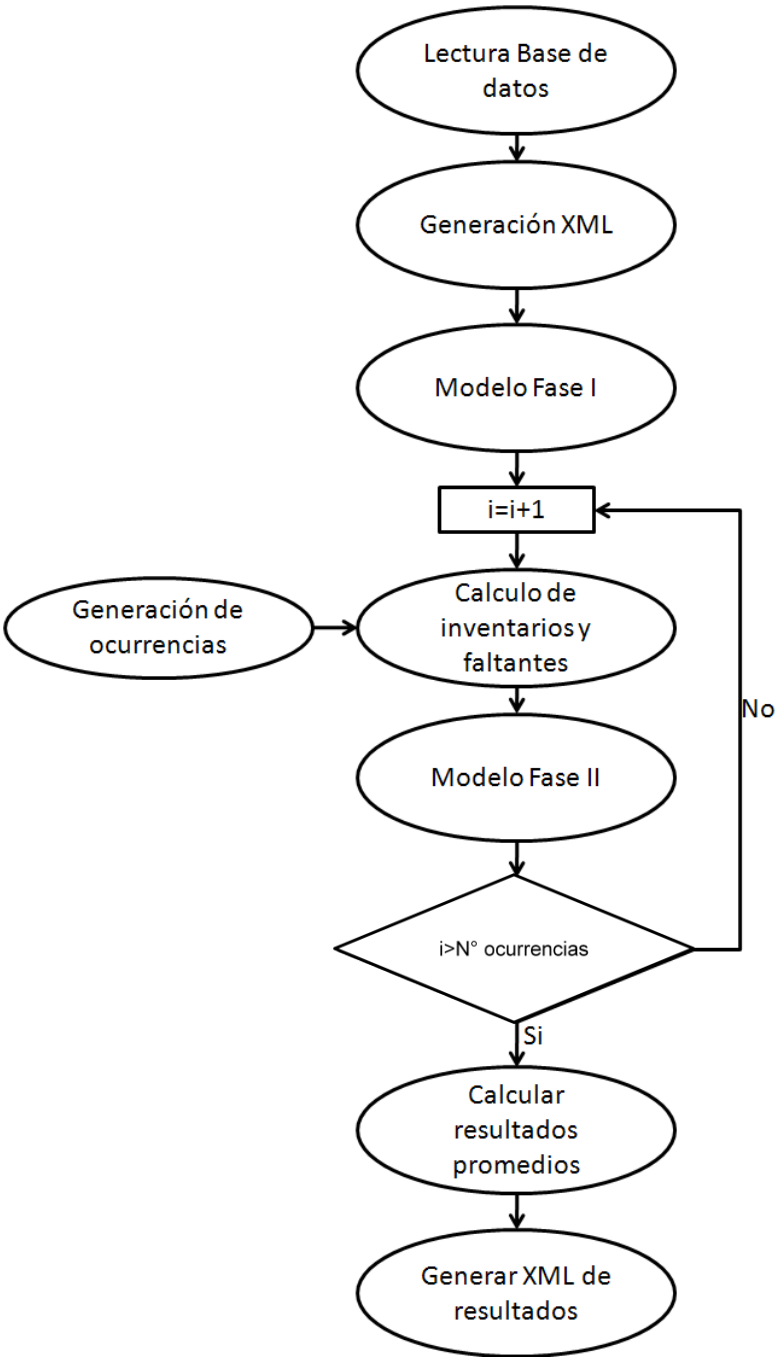


Figura 3.16: Implementación de los modelos

Como se puede observar, en la fase I se lanza el modelo estocástico (en función de varios escenarios de la demanda) y se encuentra una solución (que abarca todo el horizonte de planificación) propuesta como óptima para dichos escenarios. Luego se entra en un bucle en donde se generan las diferentes simulaciones de la

demanda (generación de ocurrencias) y se calcula, para cada demanda, los inventarios y pedidos pendientes desde $t = 0$ hasta $t = \tau\text{-}1$ que resultan de contrastar el plan de producción generado en la fase I con la demanda simulada. Esta información, más las capacidades contratadas en la fase I (de los proveedores lejanos para todo el horizonte de planificación) y las recepciones planificadas (pendientes en el momento de la segunda decisión), pasa a la fase II y se lanza el modelo de la fase II tantas veces como ocurrencias de la demanda se deseen evaluar. Cuando se han ejecutado todas las ocurrencias, se calcula el promedio de los resultados y se genera un archivo XML.

El modelo total (que abarca los modelos Fase I, Fase II y los algoritmos) se ejecuta en el instante $t = 0$, en el cual se desconoce cuánto será la demanda real de cada producto terminado, por ello el modelo total incluye la ejecución del número de ocurrencias de la Fase II que el decisor estime convenientes (porque la demanda real puede ser mayor, igual o menor que la estimada inicialmente). Es decir, los resultados del modelo total darán la solución más robusta para los distintos posibles valores de la demanda real de cada producto terminado. El decisor puede probar el modelo tanto para una sola simulación de la demanda para la segunda fase como para diez mil simulaciones de ésta. En el segundo caso los resultados serán el promedio de las diez mil ocurrencias, por lo tanto se habrá probado el modelo para cualquier demanda posible dentro los rangos de valores de los parámetros *p*, *q* y *m* del método de difusión de Bass.

Los resultados abarcan los costes de la primera etapa (de $t = 0$ a $t = \tau\text{-}1$) más el promedio de los cotes de la segunda etapa (de $t = \tau$ a $t = T$). Los costes de la capacidad contratada de los proveedores lejanos desde $t = 0$ hasta $t = T$ están incluidos en los costes totales aunque la decisión de cuánta capacidad contratar sea efectuada por el modelo Fase I.

Las funciones objetivo de los modelos Fase I y Fase II, maximizan el beneficio como diferencia entre los ingresos por ventas de productos terminados y todos los costos operativos, por lo tanto el resultado del modelo total da la decisión de mayor beneficio esperado en el instante t = 0, que es cuando se decide qué proveedores contratar, qué capacidad de los proveedores lejanos contratar (para todo el horizonte de planificación), y cuánto producir de cada producto (componentes, subensambles y

productos terminados) en qué procesos de cuáles proveedores, cuánta materia prima comprar (y dónde), qué productos transportar (de dónde a dónde y cuánto), etc., desde $t = 0$ hasta $t = \tau$.

3.7. Conclusiones

Los modelos propuestos permiten obtener soluciones óptimas y proporcionan las cantidades de productos terminados a fabricar en forma directa y en forma indirecta (con aplazamiento), asimismo las cantidades de productos intermedios y componentes a producir y transportar, y cuándo hacer el ensamble final (tanto en el proveedor lejano como en el local). Gracias a su estructura, se puede tener componentes compartidos por los distintos productos finales y se facilita su transporte y control de inventarios. También determina las cantidades de materias primas a comprar en cada periodo y para cada proveedor.

Mediante el enfoque de dos momentos de decisión, la programación estocástica bi-etapa y la simulación de la demanda se planifica la producción de cada proveedor de manera que, teniendo en cuenta los plazos de entrega, se ajuste la utilización de las capacidades para obtener el máximo beneficio total.

El uso del concepto de *stroke* facilita el modelado; permite agregar procesos y transportes, permite modelar fácilmente procesos alternativos (con distintos grados de aplazamiento), facilita la determinación del inventario, facilita la determinación de la capacidad utilizada en cada recurso, y reduce tanto el número de variables de decisión como el número de restricciones necesarias. El concepto de *stroke* es usado por primera vez en la planificación táctica de las operaciones de CS en esta tesis (y en el artículo de Calderón et al. (2009) relacionado con esta tesis).

Además, los modelos maximizan la diferencia entre los ingresos por ventas y los costes (todos los costes) de manera que deciden, ante recursos escasos, de cuál producto conviene producir mayor cantidad y de cuál menos (por esto es mejor que un modelo que sólo minimiza costes). Los modelos permiten penalizar la demanda no atendida con un coste de oportunidad (incluyéndolo en el coste de entregas diferidas el último periodo).

Además, los modelos permiten analizar diversos grados de variabilidad de la demanda y también analizar variaciones de costes y capacidades para determinar en qué condiciones compensará contratar la producción a proveedores cercanos frente a proveedores lejanos o a ambos en distintos periodos y cantidades.

Mediante el método de Monte Carlo y el modelo de difusión de Bass se simula la demanda real para productos innovadores y se consigue resultados óptimos con el modelo Fase II (determinista). Luego estos resultados son promediados para las distintas ocurrencias simuladas. De esta manera se obtiene el resultado total (Fase I y Fase II) para las distintas ocurrencias posibles de la demanda.

El modelo Fase II considera que el inventario final de los productos terminados no se venderá todo al mismo valor residual unitario sino que una parte del inventario final se venderá al valor residual y otra parte se venderá a un valor menor. El modelo Fase II permite indicar qué parte del inventario se venderá al valor residual (por ejemplo: el inventario final hasta como máximo el 10% de la demanda total de ese ítem) y también permite indicar cuál será el valor menor al que se liquidará el resto del inventario si lo hubiera (por ejemplo: el valor menor puede ser el 50% del valor residual).

La herramienta diseñada para la implementación del modelo permite la ejecución del los modelos Fase I y Fase II consecutivamente, empezando por ingresar los datos al modelo Fase I, ejecutando dicho modelo, luego alimentando el segundo modelo con los resultados del primero, simulando la demanda real para el segundo momento de decisión, ejecutando el modelo Fase II tantas veces como el decisor lo considere necesario (igual que el número de demandas simuladas), promediando los resultados y colocando en Excel los resultados en tablas. Estas tablas presentan los resultados para cada periodo y la suma total de: los beneficios, las ventas, los ingresos por remate final, los costos totales y parciales (de *strokes*, de capacidad, de inventario, de pedidos pendientes, de compras y, de sobre-utilización y sub-uso de la capacidad de cada recurso), así como los *strokes* ejecutados, la generación de productos, las cantidades de materias primas compradas por ítem y por periodo.

Los modelos se pueden ampliar para considerar: economías de escala en los costes de transporte y en la compra de materiales, costes de producción dependientes

del volumen, tamaño máximo del lote de producción, cantidad mínima de transporte, etc.

3.8. Referencias

Alonso-Ayuso, A., Escudero, L.F., Garín, A., Ortuño, M.T. y Pérez, G. (2003). An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. *Journal of Global Optimization*, 26, 97-124.

Arntzen, B.C., Brown, G.G., Harrison, T.P. y Trafton, L.L. (1995). Global Supply Chain Management at Digital Equipment Corporation. *Interfaces*, 25(1), 69-93.

Aviv, Y. y Federgruen, A. (2001). Design for postponement: a comprehensive characterization of its benefits under unknown demand distributions. *Operations Research*, 49(4), 578–598.

Bakir, A. M., y Byrne, M. D. (1998). Stochastic linear optimisation of an MPMP production planning model. *International Journal of Production Economics*, 55(1), 87-96.

Bass, F. (1969). A New Product Growth for Model Consumer Durables. *Management Science*, 15(5), 215-227.

Baykasoglu, A. (2001). MOAPPS 1.0: aggregate production planning using the multiple-objective tabu search. *International Journal of Production Research*, 39(16), 3685-3702.

Blackburn, J.D. (1991). The quick-response movement in the apparel industry: a case study in time compressing supply chains, in Blackburn, J.D. (Ed.), Time Based Competition, Business One Irwin, Homewood, IL, pp. 246-69.

Boulaksil, Y., Grunow M., y Fransoo, J.C. (2011). Capacity flexibility allocation in an outsourced supply chain with reservation. *International Journal of Production Economics*, 129(1), 111-118.

Calderón-Lama, J-L, García-Sabater, J-P, y Lario, F-C (2009). Modelo para la Planificación de Operaciones en Cadenas de Suministro de Productos de

Innovación. Operations Planning Model for Supply Chains of Innovative Products. Revista *Dyna*. 84(6), 10 pp.

Chen, Z.L.; Li, S., y Tirupati, D. (2002). A scenario-based stochastic programming approach for technology and capacity planning. *Computers & Operations Research*, 29(7), 781-806.

Chen, C.L. y Lee, W.C. (2004). Multi-objective optimization of multi-echelon supply chain networks with uncertain product demands and prices. *Computer & Chemical Engineering*, 28 (6-7), 1131-1144.

Chern, C.-C. y Hsieh J.-S. (2007). A heuristic algorithm for master planning that satisfies multiple objectives. *Computers & Operations Research*, 34(11), 3491-3513.

Christopher, M. (2000). The Agile Supply Chain Competing in Volatile Markets. *Industrial Marketing Management*, 29, 37–44.

Christopher, M., Peck, H. y Towill, D. (2006). A taxonomy for selecting global supply chain strategies. *The International Journal of Logistics Management*, 17(2), 277-287.

Escudero, L.F., Galindo, E., García, G., Gómez, E. y Sabau V. (1999). Schumann, a modelling framework for supply chain management under uncertainty. *European Journal of Operational Research*, 119 (1), 14-34.

Fisher, M. (1997). What is the right supply chain for your product? *Harvard Business Review*, 75(2), 105-117.

Fisher, M. y Raman, A. (1996). Reducing the cost of demand uncertainty through accurate response to early sales. *Operations Research*, 44(1), 87-99.

Garcia-Sabater, J. J., Cardos, M., y Garcia-Sabater, J. P. (2006). Un algoritmo para la Planificación de Producción en un Sistema en Red de Fabricación basada en Sistemas Multiagente. X Congreso de Ingeniería de Organización. Valencia, España.

Garg, A. y Tang, C.S. (1997). On postponement strategies for product families with multiple points of differentiation. *IIE Transactions* 29, 641-650.

Gong, H. (2005). Benefits of Postponement for fashion products with forecast updates. Submitted to the Engineering Systems Division in partial fulfillment of the requirements for the degree of Master of Engineering in Logistics. Massachusetts Institute of Technology.

Guiffrida, A.L., y Nagi, R. (2006). Cost characterizations of supply chain delivery performance. *International Journal of Production Economics*, 102(1), 22-36.

Gutierrez, José (2007). Suministro de alta velocidad. *Logística Integral*. Junio-Julio, pp. 39-42.

Harrison, A. y van Hoek, R. (2008). Logistics Management and Strategy: Competing Through the Supply Chain. 3rd Edition. Prentice Hall. Financial Times. Pearson Education Limited. pp. 148-149.

Jackson, J.R., y Grossmann, I.E. (2003). Temporal decomposition scheme for nonlinear multisite production planning and distribution models. *Industrial and Engineering Chemistry Research*, 42(13), 3045–3055.

Kaipia, R. y Holmström, J. (2007). Selecting the right planning approach for a product. *Supply Chain Management: An International Journal*, 12(1), 3-13

Kisperska-Moron, D. and Swierczek, A. (2011). The selected determinants of manufacturing postponement within supply chain context: An international study. *International Journal of Production Economics*. Accepted date: 9 September 2010. Artículo en prensa al 18-04-2011.

Kumar, S. y Arbi, A.S. (2008). Outsourcing strategies for apparel manufacture: a case study. *Journal of Manufacturing Technology Management*, 19(1), 73-91.

Kurawarwala, A. A., y Matsuo, H. (1996). Forecasting and Inventory Management of Short Life-Cycle Products. *Operations Research*, 44(1), 131-150.

Lee, H.L., y Billington, C. (1994). Designing products and processes for postponement. Boston. Kluwer Academic Publishers. Pp. 102-105.

Lee, H.L. y Billington, C. (1995). Evolution of supply chain management models and practice at Hewlett-Packard Company. *Interfaces* 25(5), 42–63.

Lee, H.L. (1998). Postponement for mass customization: satisfying customer demands for tailor made products. In: J. Gattorna, ed., Strategic supply chain alignment, Aldershot, UK, Gower, 77–91.

Lee, H.L. (2002). Aligning Supply Chain Strategies with Product Uncertainties. *California Management Review*, 44(3), 105-119.

Lee, Y.H., Kim, S.H. y Moon, Ch. (2002). Production-distribution planning in supply chain using a hybrid approach. *Production Planning & Control*, 13(1), 35–46.

Lejeune, M.A. y Yakova, N. (2005). On characterizing the 4 C's in supply chain management. *Journal of Operations Management*, 23(1), 81-100.

Leung, S.C.H. y Ng, W-L. (2007a). A goal programming model for production planning of perishable products with postponement. *Computers & Industrial Engineering*, 53 (3), 531-541.

Leung, S.C.H. y Ng, W-L. (2007b). A stochastic programming model for production planning of perishable products with postponement. *Production Planning & Control*, 18(3), 190-202.

Li, J. y Liu, L. (2008). Supply chain coordination with manufacturer's limited reserve capacity: An extended newsboy problem. *International Journal of Production Economics*, 112(2), 860-868.

Mahajan, V.; Muller, E.; y Bass, F. (1995). Diffusion of New Products: Empirical Generalizations and Managerial Uses. *Marketing Science*, 14(3), 79-88.

Meade, N., e Islam, T. (2006). Modelling and forecasting the diffusion of innovation - A 25-year review. *International Journal of Forecasting*, 22(3), 519-545.

Morrison, J. (1995). Lyfe-Cycle Approach to new Product Forecasting. *The Journal of Business Forecasting Methods and Systems*, 14(2), 3-5.

Peidro, D., Mula, J. y Poler, R. (2007). Supply chain planning under uncertainty: a fuzzy linear programming approach. Fuzzy Systems Conference. FUZZ-IEEE 2007. IEEE International. July 2007. Pp: 1-6.

Rohde, J. y Wagner, M. (2005). Master Planning. En Stadtler, H. y Kilger, C. eds., (2005). Supply Chain Management and Advanced Planning. Springer. pp. 159-177.

Romano, P. (2009). How can fluid dynamics help supply chain management? *International Journal of Production Economics*, 118, 463–472.

Scitovski, R., y Meler, M. (2002). Solving parameter estimation problem in new product diffusion models. *Applied Mathematics and Computation*, 127(1), 45-63.

Sohn, S.Y. y Lim, M. (2008). The effect of forecasting and information sharing in SCM for multi-generation products. *European Journal of Operational Research*, 186(1), 276-287.

Stratton, R. y Warburton, R.D.H. (2003). The strategic integration of agile and lean supply. *International Journal of Production Economics*, 85(2), 183–198.

Stratton, R. y Warburton, R.D.H. (2006). Managing the trade-off implications of global supply. *International Journal of Production Economics*, 103(2), 667–679.

Swanminathain, J.M., y Lee, H.L. (2003). Design for postponement. North Holland Publishing.

Wong, C.Y., Arlbjørn, J.S., Hvolby, H-H. y Johansen, J. (2006). Assessing responsiveness of a volatile and seasonal supply chain: A case study. *International Journal of Production Economics*, 104(2), 709-721.

Yang, B. y Yang, Y. (2010). Postponement in supply chain risk management: a complexity perspective. *International Journal of Production Research*, 48(7), 1901–1912.

4. APLICACIÓN DE LOS MODELOS A DISTINTAS PROBLEMÁTICAS DE LA CADENA DE SUMINISTRO Y ANÁLISIS

4.1. Introducción

Si bien el modelo de planificación propuesto puede ser aplicado a toda industria de productos innovadores con procesos estables, como son las CS de respuesta rápida de ropa, juguetes, etc., se ha contactado una empresa de sector de plásticos, la cual diseña y produce en Valencia e importa de China a la vez para atender el mercado español. Esta empresa nos ha proporcionado la información necesaria para aplicar los modelos.

En este capítulo se presentan tres casos elegidos para analizar distintas problemáticas de la CS: analizar la conveniencia de tener dos proveedores (uno local y el otro lejano) de los mismos productos, analizar el aplazamiento del ensamble final, y analizar un caso de varios productos finales con cuatro niveles en su lista de componentes y con componentes comunes. En los tres casos se ha hecho análisis del efecto de la variación de distintos factores como son: los costes, los plazos de entrega, la incertidumbre de la demanda, la ubicación del segundo momento de decisión, etc. Los resultados de las distintas ejecuciones de los modelos a su vez se analizan para inferir reglas o recomendaciones a las CS de respuesta rápida.

El resto del capítulo se organiza como sigue, en el segundo apartado se muestra el análisis de tener dos proveedores o uno solo, en el tercer apartado se muestra el análisis de aplazamiento, en el cuarto apartado se presenta el análisis con seis productos terminados y en el quinto apartado van las conclusiones.

4.2. Problemática del Doble proveedor (*double sourcing*)

Este es el caso más simple y consiste en tener dos proveedores capaces de producir el 100 por ciento del producto cada uno, con costes, plazos de entrega (*lead times*) y capacidades diferentes, y sin intercambio de componentes, sin modularidad ni aplazamiento (*postponement*). Ambos proveedores atienden una misma demanda.

En la figura 4.1 se muestra un esquema de esta situación: ambos proveedores producen A y lo entregan en un mismo destino (valga decir un almacén central), ambos

emplean la misma materia prima (B=C) la cual se distingue sólo por su ubicación y coste (B<C).

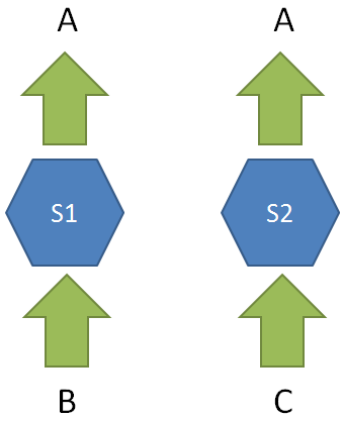


Figura 4.1: Doble proveedor (elaboración propia –**e.p.**-)

El proveedor 1 es lejano (de ultramar), compra la materia prima B en su propio país, tiene costes (de materia prima, proceso e inventario) menores que el proveedor 2, pero tiene un plazo de entrega mayor que el proveedor 2. En este caso se contrata capacidad, no producción, y se tiene costes de sobre-utilización y sub-utilización. Además este proveedor tiene el coste y tiempo de transporte en barco o en avión (transporte desde ultramar hasta el almacén central en el país de destino).

El proveedor 2 es local (es decir ubicado en el mismo país o región de destino de la mercadería, valga decir junto al almacén central), compra materia prima C en su propio país, tiene costes mayores que el proveedor 1, pero tiene un plazo de entrega menor que el proveedor 1. En este caso se contrata la producción, no la capacidad, y sólo se tiene costes de sobre-utilización si se pasa de la capacidad máxima disponible localmente. No tiene costes de transporte.

El objetivo de esta experimentación es determinar el efecto de distintos costes, diferentes grados de incertidumbre en la previsión de la demanda, distinto número de escenarios, diferentes plazos de entrega y distinta ubicación del segundo momento de decisión, sobre la conveniencia de tener uno o dos proveedores (esta experimentación se puede hacer también para un mayor número de proveedores).

Los datos usados para esta experimentación guardan relación con los costes, plazos de entrega y capacidades reales.

Los datos comunes a los dos proveedores son:

	Valores	Observaciones
Precio de venta	20 €/unidad	Igual en todos los períodos
Valor residual	10 €/unidad	Sólo en el último período
Demanda prevista	Los mismos escenarios	Diferente en cada período
Demanda real (sumatoria de la demanda de todos los detallistas)	La misma demanda para los dos proveedores (generada aleatoriamente con el método de Bass)	Diferente en cada ocurrencia (cambian m, p y q)
Número de períodos	6	Meses
Inicio de la demanda	3er. Mes	
Momento de segunda decisión	4to. Mes	
Coste de pedidos pendientes (sólo del producto terminado A)	0,0,1,1,2,3 para cada uno de los 6 periodos respectivamente	Los dos primeros periodos es cero porque no hay demanda
Coste de inventario de A	0.1 €/unidad*periodo	Igual en todos los periodos

Los datos no comunes son:

	Proveedor 1 (de ultramar)	Proveedor 2 (local)
Coste de materia prima	2 €/unidad	4
Coste de fabricación	10 €/unidad	15

Coste de inventario de B y C	0.01 €/unidad de B * periodo	0.02 €/unidad de C * periodo
Coste sobre-utilización de la capacidad	0.1 €/unidad*periodo	0.15
Coste sub-utilización	0.1 €/unidad*periodo	0
Capacidad / periodo	60,000 u/periodo	40,000
Plazo de entrega	2 periodos	1

4.2.1. Efecto de la variación de costes

Este primer grupo de ejecuciones del modelo se ha hecho con un solo escenario y una sola demanda real (generada automáticamente con el método de Bass), de manera que el modelo resulte determinista.

La demanda del producto A es:

Período	1	2	3	4	5	6
Demanda prevista	0	0	30,000	60,000	100,000	50,000
Demanda real	0	0	Se genera con Bass en cada ejecución			

Primeramente, se ha examinado el efecto de la diferencia de costes entre el proveedor lejano y el proveedor local. Se han abarcado los costes de producción (incluyendo transporte en el caso del proveedor lejano), inventario, materia prima, sobre-utilización y sub-utilización de la capacidad.

El modelo se ha ejecutado 6 veces de manera que vemos cómo varían los resultados al disminuir la diferencia de costes. Se ha considerado la diferencia de costes real actual como diferencia 100% en la primera ejecución; luego se ha disminuido dicha diferencia un 20% en cada ejecución; así en la sexta ejecución no hay ninguna

diferencia. En la tabla 4.1 se muestran los costes de fabricación (€/unidad) que se han usado en cada ejecución. En el caso del proveedor lejano, el coste de fabricación incluye el coste de transporte hasta el almacén central.

Ejecución	1	2	3	4	5	6
Proveedor lejano	10	11	12	13	14	15
Proveedor local	15	15	15	15	15	15

Tabla 4.1: Costes de fabricación (elaboración propia –**e.p.**-)

Los resultados se muestran en la tabla 4.2 y en la figura 4.2. Se puede ver como con la diferencia de precio actual toda la producción se haría en el proveedor lejano; lo mismo hasta cuando la diferencia de costes es de 40%. Para la diferencia de costes de 20% se ve que el proveedor lejano produciría el 95% mientras que el local sólo 5%.

Ejecución	1	2	3	4	5	6
Proveedor lejano	100	100	100	100	95	60
Proveedor local	0	0	0	0	5	40

Tabla 4.2: Porcentaje de la producción en cada proveedor (e.p.)

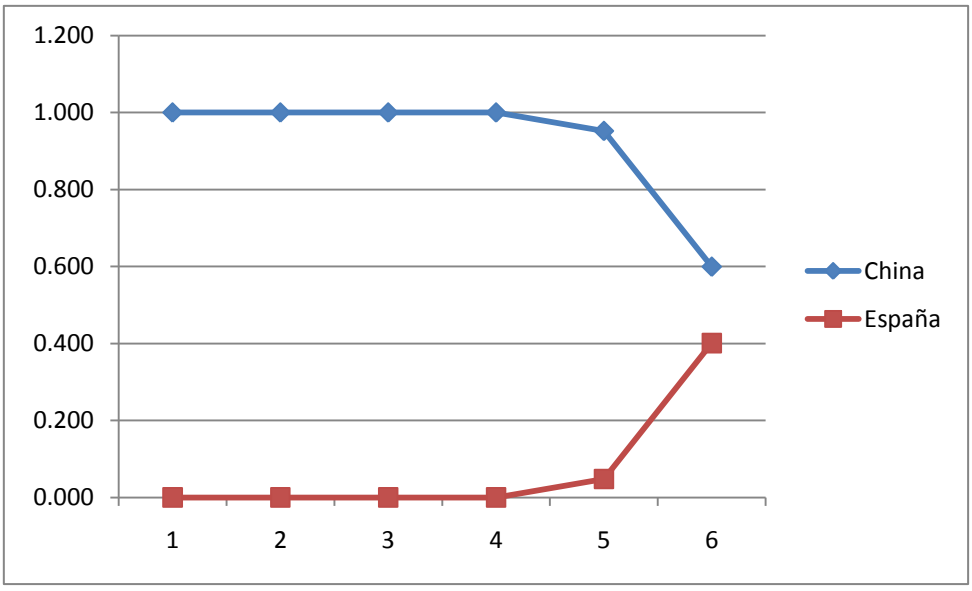


Figura 4.2: Porcentaje de la producción en función de la diferencia de costes (e.p.)

Para igualdad de costes (ejecución 6) se ve que el proveedor lejano aún producirá 60% y el proveedor local sólo 40%. Esta aparente contradicción es debida a que el proveedor lejano tiene un plazo de entrega de 2 periodos, mientras el proveedor local sólo de 1 periodo, y no se ha considerado coste de inventario de producto en proceso. Si se considera en el coste de *stroke* del proveedor lejano el coste de inventario correspondiente a la diferencia de los plazos de entrega (2-1 = 1 periodo), entonces el resultado es:

Porcentaje de la producción hecha por el proveedor lejano: 17%

Porcentaje de la producción hecha por el proveedor local: 83%

El motivo por el que no se produce localmente toda la producción es porque la capacidad del productor local no es suficiente para satisfacer la demanda y dado que los costes son iguales, el modelo matemático obtiene la solución óptima utilizando ambas capacidades.

4.2.2. Efecto de la incertidumbre y el número de escenarios

Para analizar el efecto de la incertidumbre y el número de escenarios sobre los resultados, se ejecutó el problema con alta y baja incertidumbre, y 3, 5 y 7 escenarios.

Se considera alta incertidumbre cuando la demanda total puede tener un valor dentro del rango de +/- 80% de su valor medio. Se considera baja incertidumbre cuando la demanda total puede tener un valor dentro del rango de +/- 40% de su valor medio. Esta demanda se genera por el método de Bass dentro de los rangos establecidos.

Los escenarios son datos para el modelo y estos datos son elaborados por expertos. De lo que se trata aquí no es de probar la bondad de las previsiones sino de ver en qué medida el mayor o menor número de escenarios influye en los resultados del modelo matemático de coordinación de la cadena de suministro.

En el Anexo 3 se muestran los datos de la demanda en cada escenario y sus probabilidades. Se ha cuidado que haya siempre una demanda media y que la demanda esperada (producto de las demandas por sus probabilidades) coincida con la demanda

media. Esta demanda media a su vez se ha usado como demanda más probable para generar valores de la demanda real por el método de Bass. Se han generado mil demandas reales en cada ejecución del modelo de la fase II. Los resultados se muestran en la tabla 4.3.

Incertidum-bre	Escena-rios	Beneficios	Nivel de Servicio	Prov. Lejano	Prov. Local	Valor Residual
Alta	3	12991	84.15%	0.93	0.07	1784
Alta	5	12688	84.02%	0.93	0.07	1885
Alta	7	13312	83.20%	0.92	0.08	1653
Baja	3	16020	90.73%	0.95	0.05	916
Baja	5	15974	90.66%	0.95	0.05	930
Baja	7	16109	90.19%	0.95	0.05	880

Tabla 4.3: Resultados de diferentes incertidumbres y escenarios (e.p.)

Como se ve en la tabla 4.3, con incertidumbre alta se obtienen menores beneficios que con incertidumbre baja, también el nivel de servicio es menor y el valor residual es mayor. El valor residual contribuye a los beneficios, por ello estos resultados dependen de qué valor de venta al final de la temporada tengan los productos. En cuanto al número de escenarios, no se ha encontrado una relación directa o inversa entre el número de escenarios y los beneficios, pero sí se ve que, ligeramente, a mayor número de escenarios, menor nivel de servicio y menor monto de ingresos por remate al valor residual (promedio en ambos casos).

4.2.3. Efecto de la variación del plazo de entrega

Para analizar el efecto de los distintos plazos de entrega, se ha tomado en cuenta el problema básico con tres escenarios y con una incertidumbre de +/- 50%. Con esto se ha ejecutado el modelo para: 12 periodos, inicio de la demanda en el quinto

periodo, segundo momento de decisión en el séptimo periodo y plazos de entrega del proveedor lejano de: 5, 4, 3 y 2 periodos mientras se mantenía constante el plazo de entrega del proveedor local (1 periodo). Como disminuir el plazo de entrega del proveedor lejano tiene un coste (por usar un modo de fabricación o un medio de transporte más rápido), este coste se ha incluido en los costes del proveedor lejano (dentro del coste de su *stroke*) pero aún con ellos no se llega a igualar los costes del proveedor local. Los datos se muestran en el Anexo 4. Además se han generado mil demandas reales en cada ejecución del modelo. Los resultados se muestran en la tabla 4.4.

Plazo de entrega	Benefi-cios	Nivel de Servicio	Prov. Lejano	Prov. Local	Valor Residual
5	169304	88.79%	0.73	0.27	4533
4	156193	90.87%	0.85	0.15	8893
3	113105	93.09%	0.95	0.05	12792
2	87294	99.62%	0.96	0.04	14778

Tabla 4.4: Efecto de la variación del plazo de entrega (e.p.)

Los resultados muestran que el beneficio disminuye a medida que disminuye el plazo de entrega del proveedor lejano (esto es debido a los mayores costes), el nivel de servicio en cambio aumenta y también el porcentaje de producción hecho por el proveedor lejano aumenta (estos dos últimos debido a su menor plazo de entrega).

El monto de remate al valor residual aumenta a medida que el plazo de entrega del proveedor lejano disminuye (debido a un mayor exceso de inventarios) y aumenta el riesgo debido a que depende del valor residual unitario. A mayor monto de remate menor rentabilidad de la CS (porque el VR es menor que el precio de venta).

4.2.4. Efecto de la variación del segundo momento de decisión

Para este análisis se ha tomado en cuenta el problema básico con tres

escenarios y con una incertidumbre de +/- 50%. Con esto se ha ejecutado el modelo para 12 periodos, inicio de la demanda en el quinto periodo y para diferentes ubicaciones del segundo momento de decisión: en los periodos 5, 6, 7 y 8. Vale recordar que las decisiones que se toman en el segundo momento de decisión afectan los subsiguientes periodos pero no el periodo correspondiente al segundo momento de decisión. Además se han generado mil demandas reales en cada ejecución del modelo. En la ejecución con el segundo momento de decisión ubicado en el octavo periodo se tuvo que reducir el plazo de entrega a 4 periodos para que entre dentro del horizonte de planificación. Los resultados se muestran en la tabla 4.5.

Momento de decisión	Plazo de entrega	Beneficios	Nivel de Servicio	Proveedor Lejano	Proveedor Local	Valor Residual
5	5	186016	89.17%	0.815	0.185	5627
6	5	176505	88.93%	0.772	0.228	5409
7	5	169421	89.26%	0.732	0.268	4576
8	4	168466	90.76%	0.810	0.190	8215

Tabla 4.5: Efecto de la variación del segundo momento de decisión (e.p.)

En vista de los resultados, se puede afirmar que a medida que el segundo momento de decisión se retrasa, el beneficio esperado es menor, el porcentaje de la producción del proveedor lejano disminuye y el valor residual disminuye, excepto en el último caso (segundo momento de decisión en el periodo 8) porque se tuvo que reducir el plazo de entrega a 4 periodos para mantener factible el modelo matemático dado que el horizonte es de 12 periodos. En otras palabras, conviene que el segundo momento de decisión sea lo más pronto posible para tener mayores beneficios (esto debido a que en el segundo momento de decisión se tendrá una previsión de la demanda muy precisa y cuanto más pronto se tenga ésta más periodos se abarcarán para la planificación de la segunda fase y se mejorarán las decisiones de producción).

4.3. Problemática del Aplazamiento (*postponement*) y su análisis

El objetivo de este apartado es analizar la relación entre la producción directa (sin aplazamiento de procesos) e indirecta (con aplazamiento de procesos) y las capacidades de los recursos de los proveedores en una CS.

El caso de estudio consiste en tener dos proveedores (un proveedor lejano y uno local) capaces de producir el 100 por ciento del producto terminado cada uno (así como sus componentes), con costes, plazos de entrega y capacidades diferentes, con productos modulares y aplazamiento (*postponement*), y con intercambio de componentes. Ambos proveedores atienden una misma demanda y sólo hay demanda externa de productos terminados.

En la figura 4.3 se muestra un esquema de esta situación: ambos proveedores pueden fabricar el producto terminado A, que se entrega en un mismo destino (un almacén central en el país del proveedor local) (*strokes* S1 y S5), y ambos emplean la misma materia prima (MC=ME) la cual se distingue sólo por su coste (MC<ME) y ubicación (MC en el país lejano y ME en el país local). Ambos proveedores pueden fabricar también el componente B (necesario para hacer A) (*strokes* S2 y S4); en este caso el proveedor lejano envía el componente B al proveedor local. El proveedor local, mediante el *stroke* 3 (S3), puede convertir el componente B en producto terminado A. Para mantener simple el problema un *stroke* 1 (S1) produce una unidad de A, lo mismo S3 y S5. De la misma manera, un S2 produce una unidad de B, lo mismo S4. Además, se necesita una unidad de materia prima (MC o ME) para producir una unidad de A, lo mismo para producir una unidad de B.

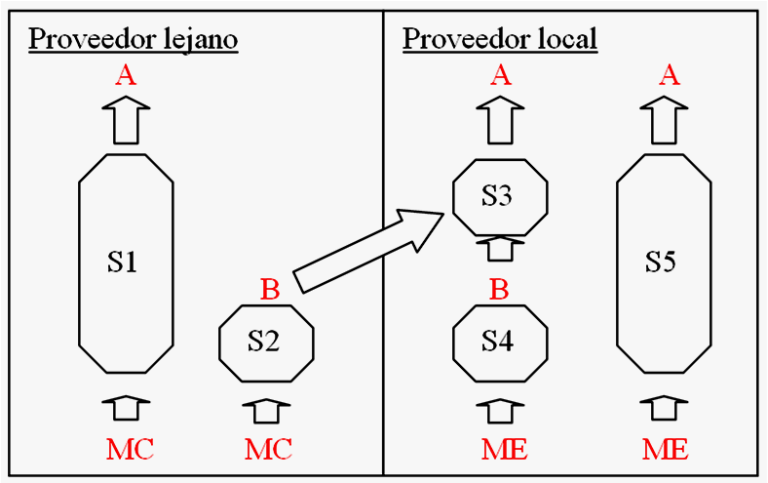


Figura 4.3: Esquema con aplazamiento (e.p.)

El proveedor lejano (de ultramar) compra la materia prima MC en su propio país, tiene costes (de materia prima, proceso e inventario) menores que el proveedor local y tiene un plazo de entrega mayor que el proveedor local. El proveedor lejano puede fabricar el producto A mediante el *stroke* 1 (S1), directamente de materia prima a producto terminado (incluyendo el transporte hasta el almacén central). Este proveedor también puede producir el componente B mediante el *stroke* 2 (S2) (incluyendo el transporte hasta el proveedor local). Para mantener la simplicidad del problema, el proveedor lejano cuenta sólo con dos recursos: R1 y R2. El S1 usa una unidad de capacidad de cada uno de los dos recursos (ver la tabla 4.6). El S2 emplea sólo una unidad de capacidad del recurso R2. El modelo matemático planifica la ejecución de los *strokes* S1 y S2, y de esta manera consume capacidades de R1 y R2. R1 y R2 tienen una capacidad máxima (que es dato del proveedor lejano). En este caso, en la Fase 1 del modelo matemático, se contrata la capacidad del proveedor lejano, capacidad de R1 y de R2, y por ello se tiene costes de capacidad contratada para todo el horizonte de planificación. En la Fase 2 se calcula sólo los costes de sobre-utilización y de sub-utilización si es que se usa más de la capacidad contratada o menos de la capacidad contratada, respectivamente. Además el proveedor lejano tiene el coste y tiempo de transporte en barco (transporte desde ultramar hasta el almacén central): el coste de transporte está incluido dentro del coste de contratación de la capacidad de R1 y R2, y el tiempo de transporte está incluido en el plazo de entrega cada *stroke*.

	R1	R2	R3	R4
S1	1	1	0	0
S2	0	1	0	0
S3	0	0	1	0
S4	0	0	0	1
S5	0	0	1	1

Tabla 4.6: Recursos que consume cada *stroke* (e.p.)

El proveedor local compra materia prima ME en su propio país, tiene costes mayores que el proveedor lejano y tiene un plazo de entrega menor que el proveedor lejano. El proveedor local puede producir directamente el producto A (mediante S5) a partir de la materia prima ME o, producir el componente B (mediante S4), almacenarlo (haciendo *postponement*) y luego transformarlo en producto terminado A (mediante S3). El proveedor local puede recibir componentes B del proveedor lejano y emplearlos para producir A (mediante S3). Para mantener la simplicidad del problema, el proveedor local cuenta sólo con dos recursos: R3 y R4. El S5 usa una unidad de capacidad de cada uno de los dos recursos (ver la tabla 4.6). El S3 emplea sólo una unidad de capacidad del recurso R3 y el S4 emplea sólo una unidad de capacidad del recurso R4. El modelo matemático planifica la ejecución de los *strokes* S3, S4 y S5, y de esta manera consume capacidades de R3 y R4. R3 y R4 tienen una capacidad máxima (que es dato del proveedor local). En el proveedor local, tanto en la Fase 1 como en la Fase 2, se contrata la producción, no la capacidad. En la Fase 2 se tiene costes de sobre-utilización si se pasa de la capacidad máxima disponible localmente, y se tiene costes de sub-utilización si no se utiliza la capacidad máxima disponible localmente. En los últimos dos periodos del horizonte de planificación se considera la capacidad máxima del proveedor local igual a cero, pues no se justifica producir en esos dos periodos (debido a que la demanda esperada –en los tres escenarios- en el último periodo es cero). El modelo matemático puede hacer producir en esos dos periodos pagando coste de sobre-utilización (esto puede suceder en las ocurrencias en las cuales la demanda simulada en la fase II sea mayor que la demanda prevista en los escenarios de la fase I). El proveedor local no tiene costes de transporte.

Los datos usados para esta experimentación guardan relación con los costes, plazos de entrega y capacidades reales.

Los datos comunes a los dos proveedores son:

	Valores	Observaciones
Número total de períodos	13	Quincenas
Precio de venta	30 €/unidad	Igual en todos los períodos

Valor residual	10 y 5 €/unidad	10 €/u hasta una cantidad menor o igual al 10% de la demanda total y luego 5 €/u para el resto de unidades.
Demanda prevista (para la Fase 1)	Tres escenarios	Con probabilidades 0.25, 0.5 y 0.25 respectivamente
Demanda real (para la Fase 2)	La misma demanda para los dos proveedores	Diferente en cada ocurrencia (cambian m, p y q)
Inicio de la demanda	5to. periodo	
Momento de segunda decisión	7mo. periodo	
Coste de *backorder* (sólo del producto terminado A)	0, 0, 0, 0, 10, 8, 6, 6, 6, 6, 6, 0.1, 0 €/unidad	Para cada uno de los 13 periodos respectivamente
Coste de inventario de A y B	0.75 €/unidad de A y 0.1 €/unidad de B	Igual en todos los periodos

Los datos no comunes son:

	Proveedor lejano	Proveedor local
Coste de materia prima	2 €/unidad	3 €/unidad
Coste de *strokes*	S1=0 y S2=0 €/unidad	S3=7.5, S4=8 y S5=15
Coste de recursos	R1=5 y R2= 5 €/unidad	R3=0 y R4=0 €/unidad
Coste de inventario de MC y ME	0.02 €/unidad de MC	0.03 €/unidad de ME

Coste sobre-utilización	R1=R2=0.5 €/unidad de capacidad	R3=3.75 €/u de capacidad R4=4 €/u de capacidad
Coste sub-utilización	R1=R2=0.5 €/unidad de cap	R3=R4=5 €/u de capacidad
Plazo de entrega	S1=5 y S2=4 periodos	S3=S4=1 y S5=2 periodos

La experimentación consiste en ejecutar el problema cambiando un parámetro cada vez y analizar los resultados. En cada ejecución se simulan tres escenarios de demanda para la fase 1 del modelo y se generan aleatoriamente mil demandas reales para la fase 2 del modelo.

4.3.1. Análisis de aplazamiento en función de la capacidad de cada proveedor

Se probaron muchas alternativas de capacidades y se eligieron las que dieron mejores resultados. En la tabla 4.7 se muestran los resultados de la experimentación. Las cuatro primeras columnas corresponden a la capacidad de cada recurso en miles de productos/periodo de tiempo (R1 y R2 corresponden al proveedor lejano, y R3 y R4 corresponden al proveedor local).

En general se puede ver que el porcentaje de producción directa[1] es muy alto (ver la columna "% Prod Directa") y el de producción indirecta[2] es muy bajo (ver la columna "% Prod Indirecta"), esto es debido a que el caso empleado para este análisis es muy sencillo (sólo un producto final con sólo un componente), de manera que no hay componentes comunes entre varios productos finales con diferentes demandas. Más adelante se mostrará el análisis con seis productos finales y sus componentes.

En las primeras cuatro filas (desde R1=10 hasta R1=40) se puede apreciar que hay una pequeña cantidad de producción indirecta, debido principalmente a que la capacidad del proveedor lejano no es suficiente para satisfacer él completamente la

[1] Producción directa: fabricación de un producto desde materia prima hasta producto terminado sin almacenamientos intermedios.

[2] Producción indirecta: fabricación de un producto por etapas, manteniendo productos intermedios en almacén hasta que sean necesarios para el ensamble final.

demanda (ver la columna "% Prov Lejano" en "De la producción total") y se debe utilizar parte de la capacidad del proveedor local (ver la columna "% Prov Local" en "De la producción total").

Tanto el proveedor lejano como el proveedor local priorizan la producción directa debido a que el coste de ésta es menor, de manera que si todos los recursos de un proveedor lejano tienen la misma capacidad, ésta se dedicará a la producción directa hasta satisfacer la demanda. En el caso del proveedor local, si todos sus recursos tienen la misma capacidad, ésta se dedicará casi completamente a la producción directa (87%[1] o más) y el proveedor local se encargará del 100% de la producción indirecta (ver la columna "% Prov Local" en "De la prod. Indirecta" en las cuatro primeras filas).

En la tabla 4.7 se puede apreciar que a mayor capacidad del proveedor lejano respecto del proveedor local, mayor porcentaje de la producción será hecha por el proveedor lejano y mayor será el beneficio[2] y la rentabilidad[3] (ver las siete primeras filas), hasta cierto punto que en el presente caso se da cuando la capacidad del proveedor lejano es de 70 mil unidades/periodo y la del proveedor local es de 20 mil unidades/periodo. En las dos últimas filas se puede ver que el beneficio y la rentabilidad disminuyen debido al mayor exceso de capacidad respecto de la demanda (es decir debido a mayores costes de sub-utilización de la capacidad local). Se puede ver, además, que si la capacidad del proveedor lejano es de 50 mil unidades/periodo o más, la producción es 100% directa. Por último, se puede apreciar, en la tabla 4.7 y en la figura 4.4 que el Nivel de Servicio[4] (NS) aumenta a medida que la capacidad total va aumentando.

Para construir al figura 4.4 se ha tenido que poner la relación entre las capacidades de los dos proveedores (C1/C2) en una escala de 0 a 1, en la cual el 1 corresponde a 4.5 (que es C1/C2 = 90/20). Lo mismo, se ha puesto en escala de 0 a 1 los beneficios (asignando 1 al mayor beneficio que es el de C1 = 70 y C2 = 20).

[1] Sale de dividir 69.19 - 8.54 de % de Prod. Indirecta entre 69.19 de % Prov. Local de la producción total, en la primera fila de la tabla 4.7

[2] El beneficio es la resta de los ingresos por ventas menos los costes totales.

[3] La rentabilidad es el cociente de beneficios entre costes totales.

[4] El nivel de servicio es el cociente del total de pedidos pendientes entre la demanda total.

Capacidades miles-u/periodo									De la producción total		De la prod. Indirecta	
R1	R2	R3	R4	Rentabilidad	Beneficio	NS	% Prod Directa	% Prod Indirecta	% Prov Lejano	% Prov Local	% Prov Lejano	% Prov Local
10	10	20	20	0,5506	269466	0,9599	91,46%	8,54%	30,81%	69,19%	0,00%	100,00%
20	20	20	20	0,5539	279139	0,9902	93,79%	5,21%	56,17%	43,83%	0,00%	100,00%
30	30	20	20	0,6560	320385	0,9912	98,69%	1,31%	76,29%	23,71%	0,00%	100,00%
40	40	20	20	0,7113	322065	0,9941	99,93%	0,07%	91,45%	8,55%	0,00%	100,00%
50	50	20	20	0,7387	332926	0,9971	100,00%	0,00%	93,68%	6,32%	0,00%	0,00%
60	60	20	20	0,7206	327145	0,9852	100,00%	0,00%	93,54%	6,46%	0,00%	0,00%
70	70	20	20	0,7583	**342482**	0,9839	100,00%	0,00%	93,30%	6,70%	0,00%	0,00%
80	80	20	20	0,7462	337748	0,9767	100,00%	0,00%	93,35%	6,65%	0,00%	0,00%
90	90	20	20	0,6974	318049	0,9539	100,00%	0,00%	94,19%	5,81%	0,00%	0,00%

Tabla 4.7: Resultados de aplazamiento en función de las capacidades de los proveedores (e.p.)

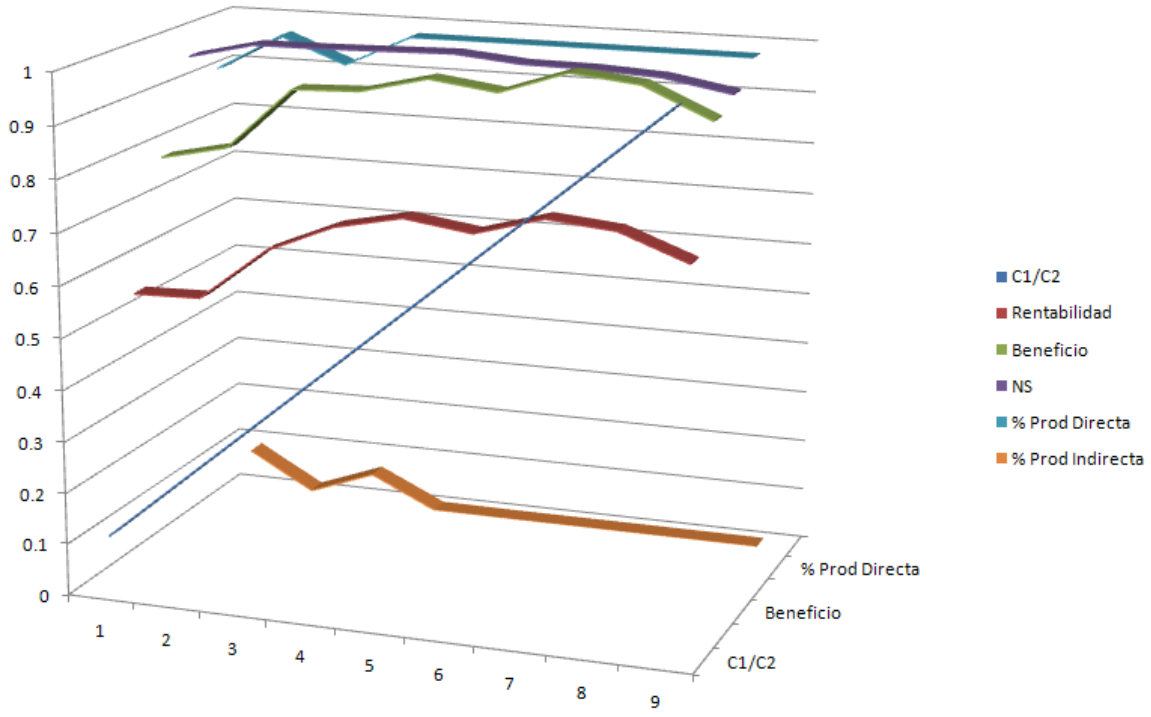


Figura 4.4: Resultados de la Tabla 4.7 (e.p.)

4.3.2. Análisis de aplazamiento en función de la capacidad de cada recurso

Se puede forzar el aplazamiento coordinando las capacidades de los recursos, es decir fijando límites a las capacidades de los recursos de manera que el proveedor lejano pueda fabricar más componentes de los que pueda ensamblar y, que el proveedor local pueda ensamblar más productos terminados que los componentes que pueda producir. Esto se consigue haciendo que el recurso R2 que produce componentes en el proveedor lejano tenga más capacidad que el recurso R1 que los ensambla, y haciendo que el proveedor local tenga más capacidad de ensamble final (recurso R3) que la del recurso R4 que produce componentes (de esta manera se puede ensamblar más en las etapas de mayor demanda a partir de componentes almacenados con anticipación).

En la tabla 4.8 se muestran seis casos con distintas capacidades máximas de R1, R2, R3 y R4. Las capacidades asignadas a los recursos corresponden a la situación de equilibrio de capacidades de manera que todos los recursos pueden ser usados al máximo. Por ejemplo, si R1 es 10 y R2 es 55 (ver la tabla 4.8) para el proveedor lejano, R1 sólo podrá ensamblar 10 componentes de los producidos por R2 y quedarán 45 componentes para enviarlos al proveedor local; a su vez R3 es 55 y R4 es 10 (en el

proveedor local), de manera que R3 puede ensamblar los 10 componentes que produce R4 y le queda capacidad para ensamblar los 45 componentes que vienen del proveedor lejano: equilibrio entre los dos proveedores.

Se puede ver en la tabla 4.8 que el mayor beneficio y rentabilidad se obtienen cuando R1=10, R2=35, R3=35 y R4=10, que es cuando la relación entre las capacidades R2/R1 es de 3.5, y también R3/R4=3.5. En este caso la relación capacidad total contratada/demanda es de 2.32 (semejante a la usada por Tavakkoli-Moghaddam et al. (2007) de 2.22). Para la relación capacidad/demanda se emplea la demanda promedio. Se puede ver, además, que el mayor beneficio y rentabilidad se obtienen cuando la producción indirecta es aproximadamente el 50% de la producción total y cuando la producción del proveedor lejano es aproximadamente el 50% de la producción total.

En la tabla 4.8 se puede ver que en las cuatro últimas columnas, la suma de porcentajes excede ligeramente al 100%, esto es debido a que se producen más componentes de los que finalmente se requieren, y queda un stock de componentes al final del horizonte de planificación. Los porcentajes se han calculado en función del número de productos terminados, por lo tanto, en estas cuatro columnas el número total de componentes excede al número total de productos ensamblados. Esto demuestra que el modelo efectivamente prevé las necesidades futuras de componentes y, como los resultados que se muestran en las tablas son promedios, en aproximadamente el 50% de los casos (cuando la demanda simulada es menor que la promedio) la producción de componentes excede a la necesaria para ensamblar productos terminados. Véase además que este fenómeno sólo sucede en los cuatro primeros casos de la tabla, que es cuando la capacidad total instalada excede la demanda total y la relación capacidad total contratada/demanda es de 2.06 o más. En las filas quinta y sexta de la tabla, no se produce componentes de más porque la capacidad total contratada/demanda es de 1.80 o menos. No se ha determinado la relación máxima de capacidad total contratada/demanda para la cual no se producirán componentes en exceso.

Capacidades miles-u/periodo									De la producción total		De la prod. Indirecta	
R1	R2	R3	R4	Rentabilidad	Beneficio	NS	% Prod Directa	% Prod Indirecta	% Prov Lejano	% Prov Local	% Prov Lejano	% Prov Local
10	55	55	10	0,5518	283467	0,9974	27,83%	72,17%	63,71%	37,83%	104,28%	0,00%
10	45	45	10	0,6244	301836	0,9952	30,69%	69,31%	63,52%	37,98%	103,94%	0,40%
10	35	35	10	0,6717	**313219**	0,9883	47,67%	52,33%	55,45%	45,46%	102,32%	1,16%
10	30	30	10	0,6575	305482	0,9679	52,50%	47,50%	51,46%	49,22%	92,30%	10,55%
10	25	25	10	0,6445	302915	0,9272	68,08%	31,92%	36,65%	63,35%	40,87%	59,13%
10	20	20	10	0,6780	301050	0,3487	68,16%	31,84%	43,02%	56,98%	80,92%	19,08%

Tabla 4.8: Resultados de aplazamiento en función de las capacidades de los recursos (e.p.)

El nivel de servicio es directamente proporcional a la capacidad total contratada (ver la figura 4.5). Se puede concluir que para cada caso real hay una relación óptima de capacidades que brindan los mayores beneficios y rentabilidad, no así nivel de servicio, pero sí un nivel aceptable de éste (excepto el sexto caso en la tabla 4.8 que tiene un nivel de servicio muy bajo).

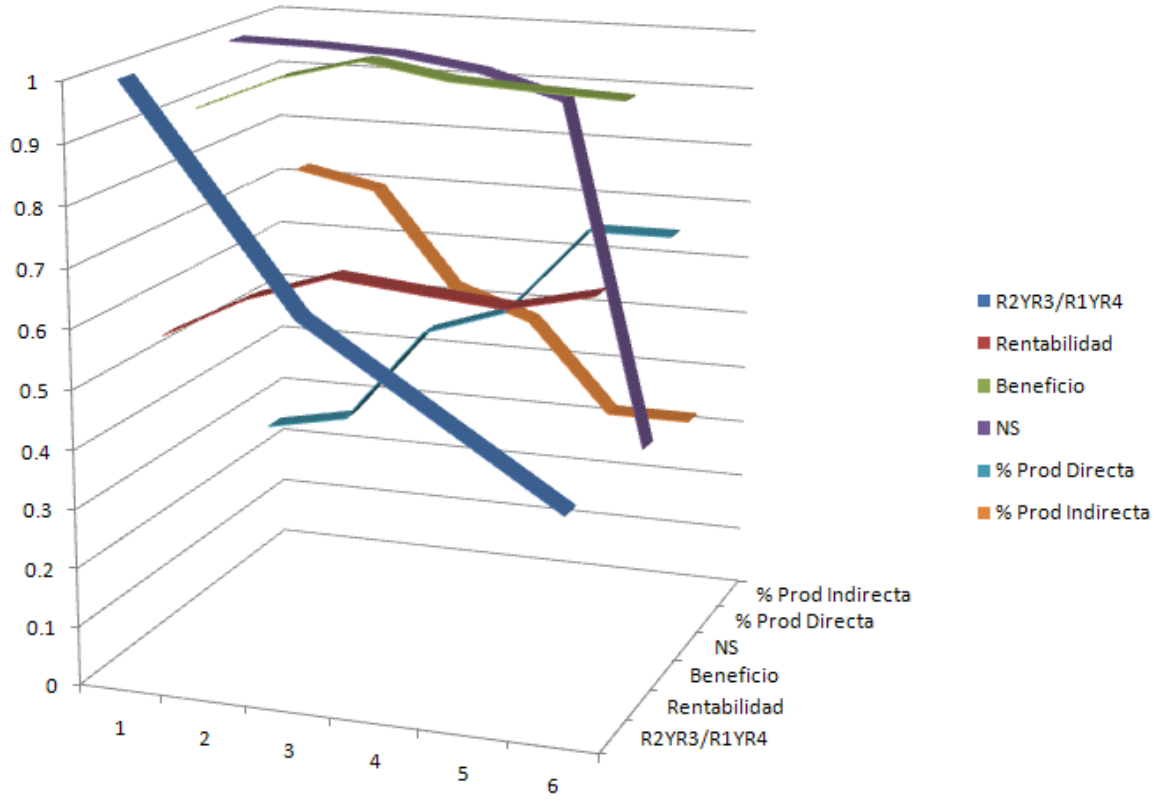


Figura 4.5: Resultados de la Tabla 4.8 (e.p.)

Para construir al figura 4.5 se ha puesto la relación entre las capacidades de los recursos (R2/R1, que es igual que R3/R4) en una escala de 0 a 1, en la cual el 1 corresponde a 55 (que es R2 = R3 = 55). Lo mismo, se ha puesto en escala de 0 a 1 los beneficios (asignando 1 al mayor beneficio que es el de R2 = R3 = 35).

Se han probado otras combinaciones de capacidad en las que no haya equilibrio de capacidades. Partiendo de la mejor combinación de la tabla 4.8, se ha generado cuatro alternativas, en todas las cuales se ha asignado al recurso R3 más capacidad de la necesaria (con la intención de dar mayor capacidad de respuesta rápida en plena temporada de ventas). Sin embargo los resultados no han sido mejores que los de la tabla anterior. En la tabla 4.9 se muestran los resultados.

Capacidades miles-u/periodo									De la producción total		De la prod. Indirecta	
R1	R2	R3	R4	Rentabilidad	Beneficio	NS	% Prod Directa	% Prod Indirecta	% Prov Lejano	% Prov Local	% Prov Lejano	% Prov Local
10	35	40	10	0,6497	307120	0,9855	47,30%	52,70%	54,92%	45,48%	100,00%	0,00%
10	30	35	10	0,6587	307936	0,9429	49,52%	50,48%	50,96%	49,51%	86,08%	15,77%
10	20	35	15	0,5861	286907	0,9830	61,09%	38,91%	40,24%	59,76%	57,68%	42,32%
10	20	30	15	0,5946	287796	0,9847	61,28%	38,72%	40,62%	59,38%	58,50%	41,50%

Tabla 4.9: Resultados dar a R3 más capacidad de la necesaria (e.p.)

Sólo se podría concluir que es mejor tener los procesos equilibrados (es decir, sus capacidades) al menos en casos sencillos como el presente de un solo producto con un solo componente.

Si bien en un principio parecería poco justificado que se fabrique un solo producto, se ha tenido conocimiento de empresas que sólo requieren un producto y/o que necesitan varios productos muy diferentes que se pueden contratar separadamente.

4.4. Problemática de varios productos y su análisis

Este caso consiste en tener dos proveedores (un proveedor lejano y uno local) capaces de producir el 100 por ciento de seis diferentes productos terminados así como sus componentes, con costes, plazos de entrega y capacidades diferentes, con productos modulares y aplazamiento (*postponement*), y con intercambio de componentes. Ambos proveedores atienden una misma demanda y sólo hay demanda externa de los productos terminados (de A1 a A6).

En la figura 4.6 se muestra un esquema de esta situación: ambos proveedores compran las materias primas y pueden fabricar los productos terminados (estos últimos se entregan en un mismo destino, un almacén central en el país del proveedor local). Los plazos de entrega de cada operación (PE) se especifican en número de periodos. No se consideran tiempos para la compra de materias primas, o en otras palabras, el plazo de entrega de la compra de materiales es de cero.

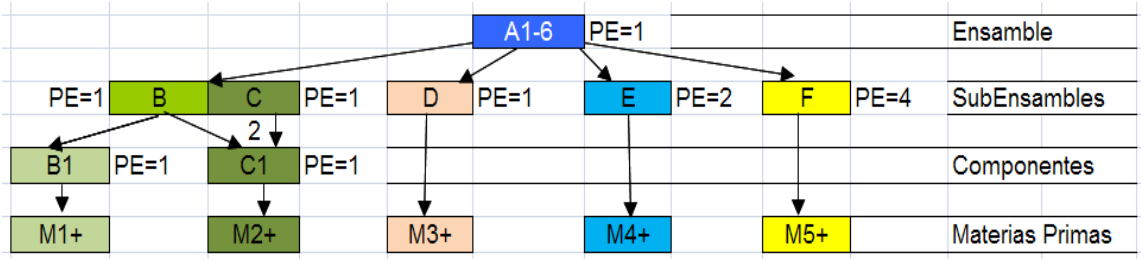


Figura 4.6: Productos y sus componentes (e.p.)

Los subensambles B y C, y los productos finales A1 a A6, se pueden elaborar tanto de forma directa como indirecta.

Los seis productos finales tienen subensambles comunes tal como se muestra en la figura 4.7.

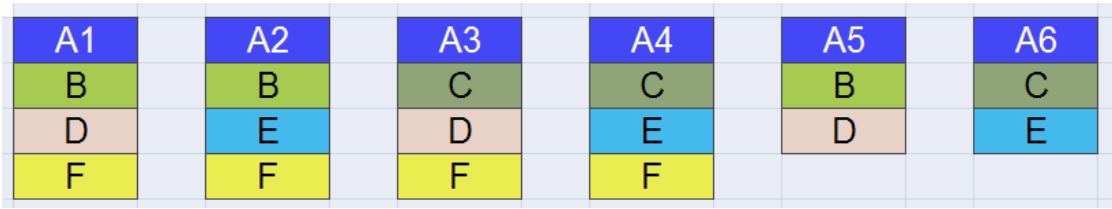


Figura 4.7: Subensambles comunes (e.p.)

Los subensambles B y C se procesan en el mismo recurso (RBC), pero B está compuesto por una unidad de B1 y una unidad de C1, mientras que C está compuesto solamente por dos unidades de C1 (ver la figura 4.8). Se considera una unidad de materia prima por cada unidad de componente a fabricar.

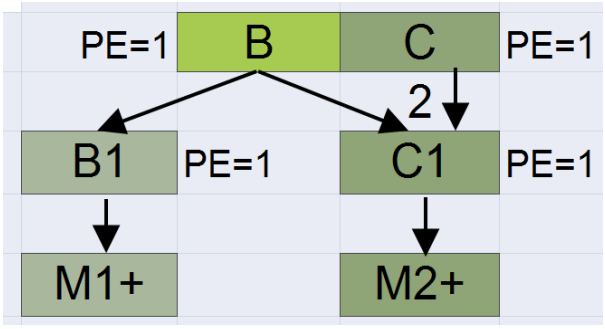


Figura 4.8: Componentes y materias primas (e.p.)

Cada proveedor tiene siete recursos, a saber:

Recurso	**Actividad del recurso**
RA	Ensamble final
RBC	Subensamble tanto de B como de C por separado
RD	Subensamble de D
RE	Subensamble de E
RF	Subensamble de F
RB1	Fabrica componentes B1
RC1	Fabrica componentes C1

Cada uno de estos recursos tendrá además una "c" si es del proveedor lejano y una "e" si es del proveedor local (por ejemplo: RFc y RFe).

Además se tienen dos recursos de transporte:

Avión	Del proveedor lejano al almacén central o al proveedor local.
Barco	Del proveedor lejano al almacén central o al proveedor local.

Cada producto terminado del proveedor lejano es enviado al almacén central en el país del proveedor local, por barco o por avión, con distintos plazos de entrega. No se almacenan productos terminados en el proveedor lejano.

Cada subensamble y cada componente del proveedor lejano pueden almacenarse en el mismo proveedor lejano o puede enviarse por avión o barco al proveedor local. Los plazos de entrega de avión y barco son dos y seis periodos (los que se deben sumar a los plazos de entrega de fabricación en cada recurso) respectivamente.

El modelo emplea *strokes* para las actividades de fabricación y transporte en los dos proveedores. Se ha modelado tanto la fabricación directa como la indirecta se ha incluido el transporte cuando se desee. Los *strokes* consumen recursos, parten de unos materiales (consumen unos ítems) y generan otros ítems. Los *strokes* tienen por lo tanto cuatro características: recursos que emplean, materiales que consumen, ítems que generan y plazo de entrega (además, en el caso del proveedor local, tienen coste pues de esa manera se costea el uso de la capacidad local).

En este caso se consideran costes de sobre-utilización y sub-utilización de la capacidad tanto en el proveedor lejano como en el local. Esto debido a que se desea analizar el efecto de variar los costes de sub-utilización. Se considera que el proveedor local tiene una capacidad máxima y que si no se usa toda se le debe pagar por la parte no utilizada. En cambio al proveedor lejano se le paga por toda la capacidad contratada y se asigna un coste de sub-utilización a discreción (se experimentan varios valores) para ver el efecto de no utilizar toda la capacidad contratada.

Los ítems tienen diferentes nombres según sus características físicas y

ubicación; por ejemplo: dos ítems iguales pero con distinta ubicación tienen distinto nombre: "Dc" es el ítem D puesto en el proveedor lejano (por ejemplo: China) mientras que "De" es el mismo ítem puesto en el proveedor local (por ejemplo: España). Los productos terminados A1 a A6 son la excepción pues como todos se entregan en el almacén central no se les ha puesto subíndice.

Los *strokes* que se envían del proveedor lejano al local tienen una "a" si se transportan por avión (por ejemplo SA1a) o una "b" si se transportan por barco. Los *strokes* que cuyo ítem generado permanece en el mismo país del proveedor tienen una "c" si es del proveedor lejano y una "e" si es del proveedor local (por ejemplo SFe). En la tabla 4.10 se pueden ver otros ejemplos (el primero, SA1b, es un *stroke* del proveedor lejano que genera el producto final A1 partiendo de materias primas del país lejano, y entrega A1 en el almacén central, incluyendo el transporte en barco).

Stroke	Ítems que consume	Recursos que emplea	Ítems que genera	Plazo de entrega	Directo o indirecto
SA1b	M1c, M2c, M3c y M5c	RAc, RBCc, RB1c, RC1c, RDc, RFc y barco	A1	11 periodos	Proceso directo
SEA1a	Bc, Dc y Fc	RAc y avión	A1	3 periodos	Proceso indirecto
SBa	M1c y M2c	RBCc, RB1c, RC1c y avión	Be	4 periodos	Proceso directo
SBBc	B1c y C1c	RBCc	Bc	1 periodo	Proceso indirecto
SDe	M3e	RDe	De	1 periodo	Proceso indirecto

Tabla 4.10: Algunos *strokes* y sus características (e.p.)

En la tabla 4.11 se muestra la relación completa de ítems, *strokes* con su plazo de entrega (PE), y recursos. Los *strokes* abarcan todas las formas posibles de producción directa e indirecta de los productos finales y de los subensambles (por ejemplo el *stroke* SBc abarca la producción directa del subensamble B (desde sus materias primas) mientas que el *stroke* SBBc abarca sólo el submontaje de B1c y C1c).

Ítems	strokes	PE	strokes	PE	Recursos
A1	SA1b	11	SEA1e	1	RAc
A2	SA2b	11	SEA2e	1	RBCc
A3	SA3b	11	SEA3e	1	RB1c
A4	SA4b	11	SEA4e	1	RC1c
A5	SA5b	9	SEA5e	1	RDc
A6	SA6b	9	SEA6e	1	REc
Bc	SA1a	7	SBa	4	RFc
Be	SA2a	7	SBb	8	RAe
Cc	SA3a	7	SBc	2	RBCe
Ce	SA4a	7	SBe	2	RB1e
Dc	SA5a	5	SBBc	1	RC1e
De	SA6a	5	SBBe	1	RDe
Ec	SA1e	5	SCa	4	REe
Ee	SA2e	5	SCb	8	RFe
Fc	SA3e	5	SCc	2	Barco
Fe	SA4e	5	SCe	2	Avión
B1c	SA5e	3	SCCc	1	
B1e	SA6e	3	SCCe	1	
C1c	SEA1b	7	SB1a	3	
C1e	SEA2b	7	SB1b	7	
M1c	SEA3b	7	SB1c	1	
M2c	SEA4b	7	SB1e	1	
M3c	SEA5b	7	SC1a	3	
M4c	SEA6b	7	SC1b	7	
M5c	SEA1a	3	SC1c	1	
M1e	SEA2a	3	SC1e	1	
M2e	SEA3a	3	SDa	3	
M3e	SEA4a	3	SDb	7	
M4e	SEA5a	3	SDc	1	
M5e	SEA6a	3	SDe	1	
	SFa	6	SEa	4	
	SFb	10	SEb	8	
	SFc	4	SEc	2	
	SFe	4	SEe	2	

Tabla 4.11: Ítems, *strokes*, PE y recursos (e.p.)

Los datos usados para esta experimentación guardan relación con los costes, plazos de entrega y capacidades reales.

Los datos comunes a los dos proveedores son:

	Valores	Observaciones
Precio de venta	50 €/unidad para A1 a A4 40 €/unidad para A5 y 44 €/unidad para A6	Igual en todos los períodos.
Valor residual	11.9 €/unidad para A1 a A4 6.6 €/unidad para A5 y A6	Hasta una cantidad menor o igual al 10% de la demanda total y luego la mitad de €/unidad para el resto de unidades.
Demanda prevista (para la Fase 1)	Tres escenarios	Con probabilidades 0.25, 0.5 y 0.25 respectivamente.
Demanda real (para la Fase 2)	La misma demanda para los dos proveedores	Diferente en cada ocurrencia (cambian m, p y q).
Número total de períodos	30	Semanas
Inicio de la demanda	Periodo 17	
Momento de segunda decisión	17mo. periodo	
Coste de pedidos pendientes (sólo de los productos terminados)	12, …, 12, 6, 0.1, 0.001 €/unidad	Para cada uno de los 14 periodos en que hay demanda.
Costes de inventario	2% del coste total unitario	Igual en todos los periodos.

Los datos no comunes son:

	Proveedor lejano	Proveedor local
Coste de materia prima	1, 2, 3, 4 y 5 €/unidad	10% más €/unidad
Coste de *strokes*	0 €/unidad	15 €/pt directo (A1 a A4) 15.2 €/pt indirecto (A1 a A4) 10.6 €/pt directo (A5 a A6)
Coste de recursos	8 €/unidad de pt	0 €/unidad
Coste de producto terminado incluyendo costes de materias primas y transporte	19.25, 20.25, 20.25, 21.25, 11.75 y 13.75 €/pt de A1 a A6 en barco respectivamente, y +0.75 €/pt en avión c/u	27.1, 28.2, 28.2, 29.3, 17.2 y 19.4 €/pt directo (A1 a A6 respectivamente) +0.2 €/pt indirecto (A1 a A4)
Coste sobre-utilización	16.67% más que el coste de producción dentro del turno	50% más que el coste de producción dentro del turno
Coste sub-utilización	10% del coste de producción dentro del turno	100% del coste de producción dentro del turno

La experimentación consiste en ejecutar el problema cambiando un parámetro cada vez y analizar los resultados. En cada ejecución se simulan tres escenarios de demanda para la fase 1 del modelo y se generan aleatoriamente cien demandas reales para la fase 2 del modelo.

4.4.1. Análisis del efecto del Valor Residual

Para este análisis se ha experimentado con distintos valores residuales y, para aquellos valores que dieron mayor beneficio total, se ha probado distintas capacidades de los proveedores, de manera que se obtenga la mejor relación de capacidades a contratar en función del valor residual esperado.

Valga recordar que no todo el inventario final es rematado al valor residual (VR). Sólo hasta el 10% de la demanda total puede ser rematada al valor residual. Si el inventario final de un producto terminado es mayor al 10% de la demanda total de dicho ítem, el número de unidades por encima del 10% se rematan un precio que es la mitad del valor residual correspondiente.

Las pruebas son:

1	VR = 80% del precio de venta En este caso el VR es mayor que el coste de producción (CP) más el coste de materias primas (CMP) en el proveedor local.	Capacidad máxima de los proveedores igual y muy grande en ambos. El modelo decide cuánta capacidad contratar por debajo de la capacidad máxima, en cada periodo.
2	VR = 80% del precio de venta	Capacidad máxima del proveedor local igual a la cuarta parte de la capacidad máxima del proveedor lejano.
3	VR = 80% del precio de venta	Capacidad máxima del proveedor local igual a la octava parte de la capacidad máxima del proveedor lejano.
4	VR = 50% del precio de venta En este caso el VR es menor que el CP+CMP del proveedor local pero mayor que el CP+CMP del proveedor lejano.	Capacidad máxima de los proveedores igual y muy grande en ambos. El programa decide cuánta capacidad contratar por debajo de la capacidad máxima, en cada periodo.
5	VR = 45% del precio de venta En este caso el VR es menor que el CP+CMP del proveedor local pero mayor que el CP+CMP del proveedor lejano.	Capacidad máxima de los proveedores igual y muy grande en ambos. El programa decide cuánta capacidad contratar por debajo de la capacidad máxima, en cada periodo.

6	VR = 40% del precio de venta En este caso el VR es menor que el CP+CMP del proveedor local pero mayor que el CP+CMP del proveedor lejano.	Capacidad máxima de los proveedores igual y muy grande en ambos. El programa decide cuánta capacidad contratar por debajo de la capacidad máxima, en cada periodo.
7	VR = 40% del precio de venta	Capacidad máxima del proveedor local igual a la cuarta parte de la capacidad máxima del proveedor lejano.
8	VR = 40% del precio de venta	Capacidad máxima del proveedor local igual a la octava parte de la capacidad máxima del proveedor lejano.
9	VR = CP+CMP del proveedor lejano de A1 (para A1 a A4) y de A5 (para A5 y A6), por ser los de menor CP+CMP, más el coste de envío en barco. Equivale al 38.5% del precio de venta.	Capacidad máxima del proveedor local igual a la octava parte de la capacidad máxima del proveedor lejano. En todos los casos los proveedores pueden hacer horas extra hasta por 50% de las horas normales.
10	VR = intermedio entre el CP+CMP del proveedor local y su CP+CMP-Cte de Sub-Utilz. Equivale al 32.6% del precio de venta.	Capacidad máxima del proveedor local igual a la octava parte de la capacidad máxima del proveedor lejano.
11	VR = CP+CMP-Cte de Sub-Utilz. del proveedor local de A4 (para A1 a A4) y de A6 (para A5 y A6), por ser los de mayor coste. Equivale al 28.2% del precio de venta.	Capacidad máxima del proveedor local igual a la octava parte de la capacidad máxima del proveedor lejano.
12	VR = valor intermedio de CP+CMP-Cte de Sub-Utilz. del proveedor local (ni el mayor ni el menor valor). Equivale al 24.8% del precio de venta.	Capacidad máxima del proveedor local igual a la octava parte de la capacidad máxima del proveedor lejano.

13	VR = CP+CMP-Cte de Sub-Utilz. del proveedor local de A1 (para A1 a A4) y de A5 (para A5 y A6), por ser los de menor coste. Equivale al 23.8% del precio de venta.	Capacidad máxima del proveedor local igual a la octava parte de la capacidad máxima del proveedor lejano.
14	VR = igual que el anterior.	Capacidad máxima del proveedor local igual que la anterior y capacidad máxima del proveedor lejano igual a la del proveedor local.
15	VR = igual que el anterior.	Capacidad máxima del proveedor local igual a la tercera parte de la capacidad máxima del proveedor lejano del caso anterior.
16	VR = igual que el anterior.	Capacidad máxima del proveedor local igual a la mitad del caso anterior. Esto es para disminuir costes de subutilización.
17	VR = 22% del precio de venta.	Capacidad máxima del proveedor local igual a la octava parte de la capacidad máxima del proveedor lejano (como en el caso 13).
18	VR = 20% del precio de venta.	Capacidad máxima del proveedor local igual a la octava parte de la capacidad máxima del proveedor lejano (como en el caso 13).
19	VR = 20% del precio de venta.	Capacidades igual a la del caso 14.
20	VR = 20% del precio de venta.	Capacidades igual a la del caso 16.
21	VR = 20% del precio de venta.	Capacidad del proveedor local nula.

Se pueden seguir analizando muchos más casos pero con estos 21 tenemos suficientes para inferir conclusiones útiles.

Los resultados son:

	Ventas totales	Valor Residual del Inventario	Costes Totales	Beneficio Esperado	Rentabilidad
1	1.18E+07	1.63E+08	1.00E+08	7.41E+07	74%
2	1.20E+07	1.04E+08	6.17E+07	5.46E+07	88%
3	1.19E+07	8.94E+07	5.18E+07	4.95E+07	96%
4	1.19E+07	6.01E+07	6.09E+07	1.11E+07	18%
5	1.19E+07	2.57E+07	4.65E+07	-8.91E+06	-19%
6	1.19E+07	1.43E+07	3.76E+07	-1.14E+07	-30%
7	1.19E+07	1.23E+07	2.14E+07	2.81E+06	13%
8	1.19E+07	1.21E+07	1.93E+07	4.68E+06	24%
9	1.20E+07	5.08E+05	7.76E+06	4.77E+06	61%
10	1.18E+07	4.35E+05	7.39E+06	4.86E+06	66%
11	1.186E+07	2.65E+05	7.25E+06	4.88E+06	67%
12	1.187E+07	9.02E+04	7.10E+06	4.87E+06	69%
13	1.20E+07	4.39E+04	7.09E+06	4.98E+06	70%
14	1.18E+07	3.75E+04	7.10E+06	4.78E+06	67%
15	1.19E+07	1.22E+05	5.85E+06	6.22E+06	106%

16	1.18E+07	1.76E+05	5.67E+06	6.32E+06	111%
17	1.16E+07	1.95E+04	6.98E+06	4.67E+06	67%
18	1.19E+07	9.99E+02	7.02E+06	4.87E+06	69%
19	1.17E+07	5.23E+02	7.04E+06	4.70E+06	67%
20	1.19E+07	2.59E+04	5.52E+06	6.42E+06	116%
21	1.18E+07	1.33E+04	5.16E+06	6.69E+06	130%

Tabla 4.12: Resultados de análisis de Valor Residual (e.p.)

En los tres primeros casos se puede ver que para el mismo VR, cuando éste es muy alto, conviene tener la mayor capacidad posible y producir lo máximo que se pueda. En el tercer caso la rentabilidad (% de BE/Costes Totales) es mayor debido a menores costes totales debido a menor capacidad máxima del proveedor local.

En el cuarto caso, con un VR=50% del precio de venta (PV), aún conviene tener la mayor capacidad posible y producir lo máximo que se pueda (este máximo tiene como límite la capacidad máxima más el máximo de sobre-utilización (50% de la capacidad máxima), pero su rentabilidad es muy baja (18%).

En el quinto y sexto caso, con un VR de 45% y de 40% del PV, el Beneficio Esperado (BE) sale negativo pues al tener una capacidad máxima muy grande, el coste de sub-utilización del proveedor local sale muy alto, y el ingreso por remate de los productos a su valor residual disminuye a la tercera y cuarta parte del cuarto caso (en el quinto y sexto caso respectivamente).

El séptimo y octavo caso, tienen VR=40% del PV, igual que el sexto caso, pero se ha reducido la capacidad máxima del proveedor local, que es la que tiene mayores costes de sub-utilización. De esta manera se ha obtenido nuevamente un BE positivo pero su rentabilidad es muy baja (13% y 24% respectivamente).

El octavo caso tiene mayor BE que el séptimo caso, tiene una capacidad

máxima local igual a la octava parte de la capacidad máxima del proveedor lejano. De aquí en adelante, para menores valores de VR se considera siempre una capacidad local menor o igual a la usada en este octavo caso.

Los casos 9 al 12 tiene un VR intermedio entre el CP+CMP del proveedor local y su CP+CMP-coste de Sub-Utilización. La capacidad máxima del proveedor local en todos estos casos es igual a la octava parte de la capacidad máxima del proveedor lejano. Se puede ver que en todos estos casos el BE es semejante.

En los casos 13 a 16, el VR es igual al CP+CMP-Coste de Sub-Utilización de A1 (para A1 a A4) y de A5 (para A5 y A6), por ser los de menor coste del proveedor local. Estos cuatro casos sólo se diferencian en la capacidad máxima de los proveedores, la cual se ha ido disminuyendo con el objetivo de reducir el coste de capacidad sub-utilizada. El de menor capacidad total, el caso 16, es el que tienen mayor BE, menores costes totales y mayor rentabilidad.

Los casos 17 a 21, son para valores de VR menores que el menor coste CP+CMP-Coste de Sub-Utilización del proveedor local, llegando hasta un VR=20% del precio de venta. En estos casos se ha ido disminuyendo la capacidad de los dos proveedores y a medida que la capacidad total era menor, mayor BE se obtenía. Los dos últimos son los que tienen mayor rentabilidad de los 21 casos analizados.

De lo anterior se puede inferir que:

- Para valores altos del Valor Residual (VR), mayores que el coste CP+CMP del proveedor local, conviene tener la mayor capacidad posible en ambos proveedores.

- Para valores de VR menores que el coste CP+CMP del proveedor local pero mayores que el coste CP+CMP-Coste de Sub-Utilización del proveedor local, conviene tener menor capacidad en ambos proveedores pero que en total satisfaga la demanda.

- Para valores de VR menores que el coste CP+CMP-Coste de Sub-Utilización del proveedor local, conviene no tener proveedor local. El proveedor lejano producirá lo necesario y a veces pecará de exceso, a veces

de defecto, pero en general los BE serán mayores que si se tuviera capacidad local y se tuviera que pagar por su sub-utilización.

4.4.2. Análisis de aplazamiento en función de la capacidad de cada recurso

Se analiza el aplazamiento para diferentes capacidades de los proveedores.

Las pruebas realizadas son:

1	Capacidad del proveedor lejano mayor que la del proveedor local.	Se toma para cada recurso la mayor cantidad de capacidad que se usó en la alternativa 9 de VR.
2	Igual que el anterior pero multiplicando por 0.5 la capacidad del ensamble final del proveedor lejano y por 2 la capacidad del ensamble final del proveedor local.	Para obligar a que se haga más ensamble final en el proveedor local.
3	Igual que el anterior pero haciendo las mismas multiplicaciones con el recurso RBC del proveedor lejano y del proveedor local.	Para obligar a que se haga más subensamble de B1 y C1 en el proveedor local.
4	Igual a la anterior pero con un mayor coste de pedidos pendientes.	Con coste de pedidos pendientes=24
5	Igual que la anterior pero reduciendo más la capacidad de los recursos RA y RBC del proveedor lejano.	Multiplicando por 0.5 las capacidades de ensamble y subensamble del proveedor lejano del caso 4 de esta serie de análisis.
6	Igual que la anterior pero con menos capacidad en todos los recursos del proveedor local.	Con la capacidad mínima necesaria en el proveedor local.
7	Igual que la anterior pero con menos capacidad de ensamble final en el proveedor lejano.	Multiplicando nuevamente por 0.5 la capacidad de ensamble final del proveedor lejano.

Los resultados obtenidos se muestran en la tabla 4.13.

	Ventas totales	Valor Residual del Inv.	Costes Totales	Beneficio Esperado	Coste de Sub-Utiliz. en el proveedor local	Prod. Directa	Prod. Indirecta
1	1.18E+07	4.34E+05	1.02E+07	1.97E+06	5.03E+06	25.1%	74.9%
2	1.19E+07	4.36E+05	1.17E+07	6.68E+05	6.43E+06	42.7%	57.3%
3	1.20E+07	4.34E+05	1.27E+07	-2.27E+05	7.37E+06	43.9%	56.1%
4	1.17E+07	4.36E+05	1.27E+07	-4.81E+05	7.45E+06	42.7%	57.3%
5	1.17E+07	4.34E+05	1.26E+07	-4.57E+05	7.29E+06	49.2%	50.8%
6	1.19E+07	4.34E+05	6.46E+06	5.89E+06	9.88E+05	48.7%	51.3%
7	1.18E+07	4.33E+05	6.62E+06	5.61E+06	9.29E+05	38.9%	61.1%

Tabla 4.13: Resultados del análisis de aplazamiento (e.p.)

Se puede ver que en los casos 1 y 2 el BE es bajo y el coste de sub-utilización en el proveedor local es un poco alto.

Los casos 3, 4 y 5 dan BE negativo debido al alto coste de sub-utilización del proveedor local.

En los casos 6 y 7 el BE es alto y el coste de sub-utilización del proveedor local es menor que los anteriores.

El caso 7, tiene BE menor que el caso 6 porque tiene más producción indirecta y más costes totales que el caso 6.

Todos estos casos han tenido como Valor Residual VR = CP+CMP del proveedor lejano de A1 (para A1 a A4) y de A5 (para A5 y A6), equivalente al 38.5% del precio de venta, y se ve que para este VR el mejor caso es el 6, el cual implica tener la mínima capacidad necesaria en el proveedor local y forzar una producción indirecta de aproximadamente 50% del total de la producción (lo cual se logra reduciendo las capacidades de los recursos de ensamble final y subensamble en el proveedor lejano).

4.4.3. Análisis del efecto de cambiar el segundo momento de decisión

Este análisis consiste en ejecutar el modelo matemático con distintos momentos para la segunda decisión de asignación de la producción.

Se prueba los siguientes momentos de segunda decisión para distintos porcentajes de incertidumbre de la demanda:

1	2do momento de decisión (2md) en el periodo 13	Con +/- 40% de incertidumbre de la demanda
2	2md en el periodo 14	Con +/- 40% de incertidumbre de la demanda
3	2md en el periodo 15	Con +/- 40% de incertidumbre de la demanda

4	2md en el periodo 16	Con +/- 40% de incertidumbre de la demanda
5	2md en el periodo 17	Con +/- 40% de incertidumbre de la demanda
6	2md en el periodo 18	Con +/- 40% de incertidumbre de la demanda
7	2md en el periodo 19	Con +/- 40% de incertidumbre de la demanda
8	2do momento de decisión (2md) en el periodo 13	Con +/- 80% de incertidumbre de la demanda
9	2md en el periodo 14	Con +/- 80% de incertidumbre de la demanda
10	2md en el periodo 15	Con +/- 80% de incertidumbre de la demanda
11	2md en el periodo 16	Con +/- 80% de incertidumbre de la demanda
12	2md en el periodo 17	Con +/- 80% de incertidumbre de la demanda
13	2md en el periodo 18	Con +/- 80% de incertidumbre de la demanda
14	2md en el periodo 19	Con +/- 80% de incertidumbre de la demanda

No se puede considerar el segundo momento de decisión más allá del periodo 19 porque el plazo de entrega más largo es de 11 periodos y daría infactibilidad.

Los resultados son:

	Ventas totales	Valor Residual del Inv.	Costes Totales	Beneficio Esperado
1	1.20E+07	8.54E+04	5.59E+06	6.49E+06
2	1.19E+07	8.99E+04	5.58E+06	6.42E+06
3	1.21E+07	9.79E+04	5.64E+06	6.57E+06
4	1.20E+07	1.42E+05	5.63E+06	6.52E+06
5	1.19E+07	1.69E+05	5.68E+06	6.40E+06
6	1.19E+07	2.13E+05	5.73E+06	6.35E+06
7	1.18E+07	2.60E+05	5.78E+06	6.26E+06
8	1.18E+07	7.82E+04	5.59E+06	6.32E+06
9	1.20E+07	8.54E+04	5.72E+06	6.37E+06
10	1.18E+07	1.37E+05	5.66E+06	6.26E+06
11	1.18E+07	1.92E+05	5.71E+06	6.27E+06
12	1.20E+07	2.05E+05	5.77E+06	6.45E+06
13	1.20E+07	2.48E+05	5.84E+06	6.42E+06
14	1.18E+07	3.19E+05	5.89E+06	6.23E+06

Tabla 4.14: Resultados del análisis del segundo momento de decisión (e.p.)

De los resultados se puede inferir que:

Para una incertidumbre de la demanda de +/- 40% conviene ubicar el segundo momento de decisión en el periodo 15 ó 16 pues son los de mayor beneficio esperado.

Para una incertidumbre de la demanda de +/- 80% conviene ubicar el segundo momento de decisión en el periodo 17 ó 18 pues son los de mayor beneficio esperado.

4.4.4. Análisis del efecto de tener mayor o menor incertidumbre

Para este análisis se toman en cuenta tres niveles de incertidumbre y dos valores residuales.

Los casos considerados son:

1	Con incertidumbre de +/- 40%	Se usa como base la versión 9 del análisis de VR
2	Con incertidumbre de +/- 60%	Se usa como base la versión 9 del análisis de VR
3	Con incertidumbre de +/- 80%	Se usa como base la versión 9 del análisis de VR
4	Con incertidumbre de +/- 40%	Se usa como base la versión 16 del análisis de VR
5	Con incertidumbre de +/- 60%	Se usa como base la versión 16 del análisis de VR
6	Con incertidumbre de +/- 80%	Se usa como base la versión 16 del análisis de VR

Los resultados son:

	Ventas totales	Valor Residual del Inv.	Costes Totales	Beneficio Esperado
1	1.20E+07	5.08E+05	7.76E+06	4.77E+06
2	1.21E+07	5.36E+05	7.78E+06	4.82E+06
3	1.20E+07	4.86E+03	7.10E+06	4.87E+06
4	1.19E+07	1.74E+05	5.68E+06	6.38E+06
5	1.18E+07	1.91E+05	5.69E+06	6.29E+06
6	1.20E+07	2.05E+05	5.77E+06	6.45E+06

Tabla 4.15: Resultados del análisis de variación de la incertidumbre de la demanda (e.p.)

Se puede ver que en los tres primeros casos (que tienen elevado valor residual y capacidad máxima grande), al aumentar la incertidumbre, manteniendo constantes todos los demás parámetros, el beneficio esperado aumenta.

Sin embargo, en los tres últimos casos, en los que el valor residual es bajo y la capacidad máxima es bastante menor (casi la imprescindible), el BE es mayor pero no establece una tendencia en función de los niveles de incertidumbre.

4.4.5. Análisis del efecto al cambiar el coste de pedidos pendientes

El coste de los pedidos pendientes (o faltantes) es un coste subjetivo y por lo tanto depende de la Dirección de la Empresa fijar su valor. Sin embargo este coste influye en el coste total de operación y por lo tanto se debe analizar su influencia en los resultados.

Para este análisis se consideran las siguientes alternativas:

1	Coste de faltantes = 6 €/ítem.periodo
2	Coste de faltantes = 12 €/ítem.periodo
3	Coste de faltantes = 24 €/ítem.periodo
4	Coste de faltantes = 32 €/ítem.periodo

Los resultados son:

	Ventas totales	Valor Residual del Inv.	Costes Totales	Beneficio Esperado	Coste de faltantes
1	1.19E+07	4.36E+05	9.98E+06	2.34E+06	1.17E+04
2	1.19E+07	4.36E+05	9.99E+06	2.31E+06	2.44E+04
3	1.18E+07	4.34E+05	9.99E+06	2.22E+06	4.75E+04
4	1.18E+07	4.36E+05	1.00E+07	2.20E+06	6.19E+04

Tabla 4.16: Resultados del análisis de coste de faltantes (e.p.)

Como se puede apreciar en la tabla de resultados, a medida que el coste unitario de faltantes es mayor, el coste total de faltantes es mayor y el BE es menor. Sin embargo el coste de pedidos pendientes afecta poco los resultados porque su participación en los costes totales es de 0.6 por ciento o menos.

4.4.6. Análisis del efecto de variar los costes discrecionales de sub-utilización

En este análisis se prueba el cambio de los costes de subutilización tanto del proveedor local como del proveedor lejano. Estos costes son discrecionales porque la dirección de la empresa les asigna un valor unitario sólo para ver el efecto de usar menos que la capacidad máxima. Las pruebas corresponden a los siguientes casos (en cada caso sólo se cambia el parámetro indicado):

1	Coste de sub-utilización del proveedor local igual a 0.5 de su coste normal.
2	Coste de sub-utilización del proveedor local igual a 0.75 de su coste normal.
3	Coste de sub-utilización del proveedor local igual a su coste normal.
4	Coste de sub-utilización del proveedor local igual a 1.25 de su coste normal.
5	Coste de sub-utilización del proveedor local igual a 1.5 de su coste normal.
6	Coste de sub-utilización del proveedor lejano igual a 1.25 de su coste normal.
7	Coste de sub-utilización del proveedor lejano igual a 1.5 de su coste normal.
8	Coste de sub-utilización del proveedor lejano igual a 0.5 del coste unitario de capacidad del proveedor lejano.
9	Coste de sub-utilización del proveedor lejano igual a su coste unitario de capacidad.
10	Coste de sub-utilización del proveedor lejano igual a 1.5 del coste unitario de capacidad del proveedor lejano.

Los resultados son:

	Ventas totales	Valor Residual del Inv.	Costes Totales	Beneficio Esperado	Coste de Sub-Utiliz. en el proveedor local o lejano
1	1.19E+07	1.51E+05	7.26E+06	4.81E+06	2.52E+06
2	1.18E+07	2.33E+05	8.57E+06	3.48E+06	3.73E+06
3	1.15E+07	4.34E+05	9.88E+06	2.07E+06	4.93E+06
4	1.19E+07	1.06E+06	1.13E+07	1.62E+06	4.17E+06
5	1.18E+07	2.48E+06	1.27E+07	1.58E+06	3.03E+06

6	1.19E+07	4.35E+05	1.00E+07	2.35E+06	4.33E+03
7	1.20E+07	4.36E+05	1.00E+07	2.39E+06	4.73E+03
8	1.18E+07	4.84E+05	1.00E+07	2.24E+06	8.67E+03
9	1.19E+07	5.14E+05	1.01E+07	2.40E+06	2.45E+03
10	1.19E+07	5.15E+05	1.01E+07	2.39E+06	2.21E+03

Tabla 4.17: Resultados del análisis de costes de sub-utilización (e.p.)

En los cinco primeros casos se puede ver que el BE disminuye a medida que el coste de sub-utilización del proveedor local aumenta. El coste total aumenta debido al incremento del coste de sub-utilización del proveedor local.

En los cinco últimos casos, en los que se aumenta el coste unitario de sub-utilización del proveedor lejano, se ven pequeños cambios en el BE sin establecer una tendencia. Las cifras de la última columna muestran los costes de sub-utilización del proveedor lejano y se puede ver que primero aumentan pero luego disminuyen debido que el modelo matemático optimiza su uso.

4.5. Conclusiones

Los modelos de planificación (Fase I y II) propuestos han sido aplicados a una empresa real, y se ha analizado distintas problemáticas típicas de las cadenas de suministro, y se ha experimentado con la variación de los factores que consideramos más importantes para la coordinación de la producción y de la capacidad entre proveedores y fabricantes de una CS.

En el análisis de doble proveedor se ha llegado a las siguientes conclusiones:

- Cuando el coste total del proveedor lejano (costes de producción más transporte) es menor al 80% del coste total proveedor local, conviene subcontratar toda la producción al proveedor lejano.

- Cuando el coste total del proveedor lejano (costes de producción más transporte) es 80% del coste total proveedor local, conviene subcontratar al proveedor lejano 95% de la producción total.

- Cuando el coste total de producción y transporte del proveedor lejano es igual que el del proveedor local, excepto por el coste de inventario durante el plazo de entrega del proveedor lejano, conviene subcontratarle el 60% de la producción.

- Cuando el coste total de producción, transporte e inventario del proveedor lejano es igual que el del proveedor local, conviene producir todo localmente o al menos usar toda la capacidad local y sólo subcontratar la producción que exceda a la capacidad local.

- Con incertidumbre alta se obtienen menores beneficios que con incertidumbre baja, también el valor residual del inventario final (los ingresos por remate de los productos sobrantes) es mayor pero el nivel de servicio es menor.

- En cuanto al número de escenarios, no se ha encontrado una relación directa o inversa entre el número de escenarios y los beneficios, pero sí se ve que a mayor número de escenarios, menor nivel de servicio y valor residual (ingresos por remate del inventario final).

- El beneficio aumenta a medida que aumenta el plazo de entrega del proveedor lejano, el nivel de servicio en cambio disminuye y también el porcentaje de producción hecho por el proveedor lejano disminuye. El valor residual también disminuye.

- A medida que el segundo momento de decisión se retrasa, el beneficio esperado es menor, el porcentaje de la producción del proveedor lejano disminuye y el valor residual disminuye. En otras palabras, conviene que el segundo momento de decisión sea lo más pronto posible siempre y cuando se conozca en ese momento la demanda subsiguiente hasta el final del horizonte de planificación.

En el análisis de aplazamiento se ha llegado a las siguientes conclusiones:

- Tanto el proveedor lejano como el proveedor local priorizan la producción directa debido a que el coste de ésta es menor, de manera que si todos los recursos de un

proveedor lejano tienen la misma capacidad, ésta se dedicará 100% a la producción directa. En el caso del proveedor local, si todos sus recursos tienen la misma capacidad, ésta se dedicará casi completamente a la producción directa (87% o más) y el proveedor local se encargará del 100% de la producción indirecta.

- Si en cada proveedor todos sus recursos tienen la misma capacidad, existe una relación óptima entre las capacidades de los proveedores que da el mayor beneficio total. Para los datos del caso analizado esta relación fue de 3.5 (capacidad del proveedor lejano entre capacidad del proveedor local); por lo tanto para cada caso particular se puede llegar a determinar la relación entre capacidades que da el mayor beneficio total.

- Si se fuerza el aplazamiento mediante la contratación de distintas capacidades de los recursos de los proveedores, contratando menos capacidad de ensamble que de producción de componentes en el proveedor extranjero y viceversa en el proveedor local, se obtiene distintos porcentajes de producción directa e indirecta. Se puede, en este caso, obtener la relación entre capacidades de los recursos y entre la capacidad total contratada y la demanda, de mayor beneficio total. En el caso analizado, se ha probado distintas situaciones en las que las capacidades de los recursos del proveedor lejano están en equilibrio[1] con las capacidades de los recursos del proveedor local, es decir se complementan perfectamente. En estas situaciones, el mayor beneficio y rentabilidad se obtienen cuando la producción indirecta es aproximadamente el 50% de la producción total y cuando la producción del proveedor lejano es aproximadamente el 50% de la producción total.

- Se han probado otras combinaciones de capacidad en las que no haya equilibrio de capacidades. Sin embargo los resultados no han sido mejores que los del caso anterior. Se puede concluir que es mejor tener los procesos equilibrados (es decir, sus capacidades) al menos en casos sencillos como el analizado de un solo producto con un solo componente.

[1] El equilibrio se refiere a que la suma de las capacidades de los proveedores (lejano y local), para cada recurso, coincide con la capacidad total necesaria para satisfacer la demanda.

En el análisis de seis productos se ha llegado a las siguientes conclusiones:

- Para valores altos del Valor Residual unitario (VR), mayores que el Coste de Producción más el Coste de Materias Primas (CP+CMP) del proveedor local, conviene tener la mayor capacidad posible en ambos proveedores.

- Para valores de VR menores que el coste CP+CMP del proveedor local pero mayores que el coste CP+CMP-Coste de Sub-Utilización del proveedor local, conviene tener menor capacidad en ambos proveedores pero que en total satisfaga la demanda.

- Para valores de VR menores que el coste CP+CMP-Coste de Sub-Utilización del proveedor local, conviene no tener proveedor local. El proveedor lejano producirá lo necesario y a veces pecará de exceso, a veces de defecto, pero en general los BE serán mayores que si se tuviera capacidad local y se tuviera que pagar por su sub-utilización.

- Para un VR equivalente al CP+CMP del proveedor lejano más el coste de envío en barco, el mejor beneficio se obtiene con la mínima capacidad necesaria en el proveedor local y forzando una producción indirecta de aproximadamente 50% del total de la producción (lo cual se logra reduciendo las capacidades de los recursos de ensamble final y subensamble en el proveedor lejano, y aumentando los mismos en el proveedor local). Este beneficio es mayor que si no se forzara el aplazamiento.

- Para una incertidumbre de la demanda de +/- 40% conviene adelantar el segundo momento de decisión para obtener un mayor beneficio, pues se logra disminuir los costes totales.

- Para una incertidumbre de la demanda de +/- 80%, considerando los mismos costes del punto anterior, conviene ubicar el segundo momento de decisión dos periodos más adelante (si con +/- 40% de incertidumbre convenía el periodo 15 ó 16, con +/- 80% de incertidumbre conviene el periodo 17 ó 18).

- Cuando se tiene un VR equivalente al CP+CMP del proveedor lejano y capacidad máxima grande, al aumentar la incertidumbre, manteniendo constantes todos los demás parámetros, el beneficio esperado aumenta. Sin embargo, cuando el VR es

bajo (equivalente al CP+CMP-Cte de Sub-Utilz. del proveedor local) y la capacidad máxima es bastante menor (casi la imprescindible), el beneficio esperado es mayor pero no establece una tendencia en función de los niveles de incertidumbre.

- A medida que el coste unitario de pedidos pendientes es mayor, el coste total de faltantes es mayor y el beneficio esperado es menor. Sin embargo el coste de pedidos pendientes afecta poco los resultados porque su participación en los costes totales es de 0.6 por ciento o menos.

- El beneficio esperado disminuye a medida que el coste de sub-utilización del proveedor local aumenta.

- Cuando aumenta el coste unitario de sub-utilización del proveedor lejano, se ven pequeños cambios en el beneficio esperado sin establecer una tendencia. A medida que aumenta el coste unitario de sub-utilización del proveedor lejano, el coste total de sub-utilización del proveedor lejano aumenta pero luego disminuye debido que el modelo matemático optimiza su uso.

Finalmente, como los análisis anteriores han sido para casos concretos con datos específicos, se puede afirmar que se ha encontrado valores de parámetros e incertidumbre para los cuales tomar acciones, si bien estos valores dependerán de cada cadena de suministro concreta y sus correspondientes datos.

4.6. Referencias

Tavakkoli-Moghaddam, R., Rabbani, M., Gharehgozli, A.H., y Zaerpour, N. (2007). A Fuzzy Aggregate Production Planning Model for Make-to-Stock Environments. Proceedings of the 2007 IEEE IEEM. Pp. 1609-1613.

5. CONCLUSIONES Y LÍNEAS FUTURAS DE INVESTIGACIÓN

5.1. Conclusiones

La competencia entre empresas y los bajos costes de producción en países de Asia, del este de Europa, de África y de Centro y Sur América, han forzado a las empresas de los países desarrollados a subcontratar toda o parte de su producción a proveedores lejanos, especialmente aquellos procesos intensivos en mano de obra. Cuando se trata de productos innovadores, la coordinación de la producción con los proveedores lejanos implica contratar capacidad de producción con mucha anticipación al inicio de la temporada de ventas (cuando la incertidumbre de la demanda es alta), sin embargo permite un segundo momento de coordinación cuando empieza la temporada de ventas, que es cuando se puede obtener una previsión más precisa de la demanda.

Como la gran interrogante de esta tesis es ¿cómo mejorar la planificación táctica de las operaciones de las cadenas de suministro de respuesta rápida *(responsive)* con estructura alternativa de procesos?, en el capítulo dos se hace un estado del arte a partir del cual se identifican las diferentes estrategias de las cadenas de suministro (CS) y se establece en qué circunstancias se debe emplear cada una; además se determina cómo se realiza actualmente la planificación táctica de las operaciones de la cadena de suministro. En el capítulo tres, se diseña un modelo matemático que mejora la planificación táctica de las operaciones de las CS (abarcando los proveedores y el fabricante) de productos innovadores, y en el capítulo cuatro se analiza distintas problemáticas típicas de las CS de respuesta rápida con estructura alternativa de procesos.

En esta tesis se aporta un modelo matemático de programación estocástica bi-etapa con un tratamiento diferente para los dos momentos de decisión, que como ya se mencionó, mejora la planificación táctica de las operaciones de las CS de respuesta rápida con estructura alternativa de procesos. En el primer momento se establece la previsión de la demanda mediante escenarios de la misma y el modelo Fase I optimiza la producción para el conjunto de escenarios con sus probabilidades. Para el segundo momento de decisión se propone un modelo determinista, modelo Fase II, que se ejecuta tantas veces como distintas demandas simuladas quiera abarcar el decisor. Las demandas se simulan mediante el método de Monte Carlo siguiendo el modelo de difusión propuesto por Bass. Según las demandas, los plazos de entrega, los costes de producción, etc., el modelo genera

el plan de producción óptimo que incluye la mezcla de productos, los aplazamientos óptimos (y en este sentido decide cuánto producir en forma directa y en forma indirecta) y qué parte de la demanda atender desde la producción del proveedor lejano y desde la producción local.

Uno de los principales aportes de esta tesis es el empleo del concepto de *stroke* en los modelos matemáticos (Fase I y Fase II) para la planificación de las operaciones de la CS. De este modo se pueden programar las actividades en vez de los productos, y con esto se simplifica la planificación táctica de la producción de múltiples productos y componentes (con muchos niveles en la lista de materiales), con varios proveedores y varias alternativas de proceso. Mediante los *strokes* se puede modelar fácilmente las distintas alternativas de producción directa e indirecta de un producto, ya sea haciendo cada proceso por separado y manteniendo inventarios intermedios, ya sea agregando la fabricación de componentes y el subensamble, ya sea agregando el subensamble y el ensamble final, ya sea agregando el ensamble y el transporte, o agregando todas las actividades desde la compra de materias primas hasta la entrega del producto terminado en una sola operación. El *stroke*, que incluye todo tipo de procesos y transportes, y que identifica los productos no sólo por sus características (ítem específico) sino también por su ubicación, facilita la asignación de costes y disminuye el número total de variables de decisión y de restricciones del modelo matemático.

Otra importante ventaja del modelo con *strokes* frente a los modelos existentes (publicados hasta la fecha) es que el modelo con *strokes* además de determinar la capacidad que conviene contratar de cada proveedor en cada periodo de tiempo, permite pagar por capacidad y pagar por producción. Este modelo permite pagar por la capacidad de los recursos contratados a proveedores lejanos y permite pagar por producto fabricado a los proveedores locales. En ambos casos también permite pagar horas extra y tiempo ocioso, dependiendo del tipo de contrato con cada proveedor.

Otro aporte importante de esta tesis es la herramienta de implementación del modelo total (es decir los modelos Fase I y Fase II) que incluye un simulador de la demanda (por métodos de difusión y no por métodos de series temporales) y varios algoritmos que gestionan la información de entrada, salida e interfases. El ingreso de datos se hace mediante hojas de Excel, estos datos se transfieren a un archivo XML de manera que sea fácil su aplicación en cualquier empresa real tomando los datos de sus bases de

datos. Luego se alimenta de datos y ejecuta el modelo Fase I en un optimizador comercial. Con sus resultados se fija la capacidad contratada de los proveedores lejanos para todo el horizonte de planificación, y se determinan las cantidades a producir de cada ítem en cada proveedor (y de transportar) hasta el segundo momento de decisión. Luego un algoritmo genera la demanda para todo el ciclo de ventas mediante el método de difusión de Bass y la simulación de Monte Carlo. Se generan tantas ocurrencias de la demanda como lo desee el decisor. Con esta demanda, otro algoritmo calcula los inventarios, pedidos pendientes y recepciones programadas en el momento de la segunda decisión tomando en cuenta la producción programada por el modelo Fase I. Luego se alimenta con estos datos el modelo Fase II y se ejecuta tantas veces como ocurrencias de la demanda se hayan generado. Los distintos resultados son promediados y transferidos a un archivo XML. Finalmente otro algoritmo transfiere los resultados a hojas de Excel.

El modelo total propuesto permite analizar diversos grados de incertidumbre de la demanda, variaciones de costes y capacidades, variaciones en los plazos de entrega, variación en la ubicación del segundo momento de decisión, para determinar en qué condiciones compensará contratar la producción a proveedores cercanos frente a proveedores lejanos o a ambos en distintos periodos y cantidades.

El modelo total permite además evaluar varias alternativas de aprovisionamiento (desde distintos proveedores en distintos países) y, en función de su solución, elegir con qué proveedores efectuar contratos mensuales, trimestrales o anuales para satisfacer la demanda interna de componentes y la demanda externa de productos terminados. De esta manera se puede configurar en el medio plazo la cadena se suministro.

El modelo también permite el análisis de varios productos con diseño modular, de manera que teniendo componentes comunes se determine cuánto y cuándo producirlos, y cuándo realizar los procesos de diferenciación (ensamble) de cada producto final. En función de la incertidumbre en la previsión de la demanda, el coste de la capacidad local y lejana, y del valor residual, se puede establecer qué cantidades de componentes producir y cuándo, para luego realizar el ensamble cuando la incertidumbre sea menor o se tenga plena certidumbre de las cantidades demandadas.

El modelo permite analizar distintas capacidades de cada recurso (de cada proveedor) para determinar las capacidades a contratar para tener distintos grados de

aplazamiento (de transporte, de subensambles y de ensamble final) y, en este sentido, balancear las capacidades para tener el menor coste total de todo el horizonte de planificación. Por ejemplo se puede analizar la conveniencia de tener en los proveedores lejanos mayor capacidad de fabricación de componentes y menor capacidad de ensamble final, a la vez que tener en el proveedor local menor capacidad de fabricación de componentes y mayor capacidad de ensamble final.

En el capítulo cuatro se ha analizado tres tipos de problemática de las CS, experimentando con distintos valores de los costes, plazos de entrega, capacidades, valor residual, número de escenarios, grado de incertidumbre de la demanda, ubicación del segundo punto de decisión y costes de sobre y sub utilización de la capacidad. Se puede concluir que para distintos valores de estos parámetros se deben tomar diferentes decisiones pero estas decisiones dependen de cada caso específico, es decir de los datos propios de cada cadena de suministro.

Los resultados del modelo total (capacidades a contratar, cantidades a producir, procesos a emplear, etc.) son los más robustos para la primera etapa (desde el $t = 0$ hasta el segundo momento de decisión) dado que el modelo total se ejecuta en el tiempo cero, al inicio del horizonte de planificación, tomando en cuenta distintos escenarios de la demanda y tomando en cuenta las ocurrencias de la demanda simuladas para el segundo momento de decisión (tantas como el decisor estime posibles en la vida real).

En el segundo momento de decisión, el decisor (en la vida real) ejecutará nuevamente sólo el modelo Fase II con la nueva previsión de la demanda que para entonces será mucho más precisa, y podrá reprogramar, para la segunda etapa (desde el segundo momento de decisión hasta el final del horizonte de planificación) la producción, el uso de capacidades, etc., de manera que el beneficio sea el máximo.

El modelo diseñado es particularmente útil para la Planificación de Operaciones CS de respuesta rápida (*responsive*), es decir la estrategia adecuada para productos innovadores con procesos estables. Sin embargo este modelo es aplicable a las CS que producen productos funcionales, intermedios e innovadores a la vez, con plazos de entrega largos (proveedores lejanos) y cortos (proveedores locales) a la vez.

El empleo de herramientas de optimización como la diseñada en esta tesis doctoral permite mejorar la coordinación de la producción, de la capacidad y del transporte entre las empresas proveedoras y fabricantes de una cadena de suministro.

5.2. Líneas futuras de investigación

Se pueden proponer diversas líneas futuras de investigación, como las siguientes:

1) Incluir en el estudio la incertidumbre de los tiempos de transporte en barco y avión.

2) Dado que también existe un riesgo de falla de los proveedores lejanos (retrasos en los procesos), debe estudiarse cómo disminuir la falla de los proveedores lejanos mediante una mejor colaboración (no sólo coordinación sino además colaboración en cuanto a conocer sus procesos, ayudarse a mejorarlos, alta fluidez en las comunicaciones, compartir objetivos y un trato de igualdad).

3) Como entre el primer y segundo momento de decisión la incertidumbre de la demanda es alta, el riesgo es elevado, se podría analizar la conveniencia de establecer un límite superior a la parte de la demanda total de cada artículo a producir antes del segundo momento de decisión (es decir, aplicar la estrategia Base y Surge).

4) En las empresas que tienen productos funcionales además de los innovadores, se podría analizar la conveniencia de fabricar sólo los productos funcionales y los componentes comunes en la primera etapa (antes del segundo momento de decisión), para luego hacer el ensamble contra pedido de los productos innovadores en la segunda etapa (después del segundo momento de decisión en plena temporada de ventas).

5) Se podría también analizar en un futuro la influencia de tener tres o más momentos de decisión y cuándo.

6) Se puede incluir en los modelos otros parámetros y variables para abarcar:

   - Tamaño mínimo del lote de producción y de la cantidad a transportar.
   - Tamaño mínimo del lote de compra.

- Capacidad mínima de contratación.
- Cantidad estándar de transporte y transportar sólo múltiplos de ésta.
- Costes de transporte unitario variables en función del volumen a transportar.
- Tres o más valores residuales para un mismo producto terminado en función de la cantidad sobrante.
- Limitar los pedidos pendientes para cada producto, en cada periodo, a un porcentaje de la demanda del mismo producto en el mismo periodo.

7) Otro aspecto que se puede considerar es el de los tipos de cambio, de manera que los costes de los productos producidos por proveedores extranjeros se manejen con incertidumbre.

8) Considerar políticas de ventas tales como "Evitar que haya productos sobrantes" a pesar de que puedan tenerse faltantes (escasez) altos.

9) Considerar la posibilidad de tener pedidos pendientes de subensambles y componentes en todos los proveedores.

10) Considerar la posibilidad de tener demanda externa de subensambles y componentes.

11) Considerar la posibilidad de tener varios almacenes de productos terminados en distintas ubicaciones, cada uno con una demanda propia de bienes, y considerar las relaciones entre estos y cada uno de los proveedores.

12) Además se puede ampliar el modelo para que incluya distribuidores y detallistas, pues la demanda en última instancia la atienden los detallistas, y se requiere que ellos determinen mejor sus necesidades de reposición. En este caso será necesario introducir la gestión de inventarios, lo cual requerirá determinar los inventarios de seguridad óptimos y el sistema de reposición más apropiado por producto y nodo (el cual además deberá variar con el tiempo y a medida que se aproxime y llegue a la temporada de ventas).

13) Se puede profundizar en la gestión de las capacidades, considerando operarios polifuncionales, de manera que en los periodos de mayor demanda los operarios de

procesos iniciales puedan pasar a procesos finales manteniendo la fuerza laboral total constante. En este caso es preciso considerar los costes de instalaciones paradas temporalmente debido a la reubicación del personal.

14) El modelo, con dos momentos de decisión, en la fase II, en cada ejecución considera una nueva previsión de la demanda y resuelve el programa lineal obteniendo una solución óptima para esa previsión. En este sentido el modelo "no sabe" en la fase I que va a haber un cambio en la previsión de la demanda en la fase II. Se deberá investigar las posibilidades de aplicación de técnicas de inteligencia artificial, como redes neuronales, para generar un modelo que "aprenda" de ejecuciones pasadas y mejore sus resultados.

15) Otras líneas de investigación incluirían definir los mejores modos de modelar la incertidumbre de un proceso de difusión antes de determinar a qué modelo se ajustan mejor, o incorporar la incertidumbre del corto plazo mediante métodos de programación difusa (*fuzziness*) en el modelo determinista (Fase II).

Anexo 1

Modelo matemático propuesto por Calderón et al. (2008)

XII Congreso de Ingeniería de Organización
2nd International Conference on Industrial Engineering and Industrial Management
Burgos, 3-5 de Septiembre de 2008

Modelo de Análisis para la Planificación y Selección de Proveedores Industriales de Productos de Innovación con Procesos Establecidos en Cadenas de Suministro[1]

José-Luis Calderón-Lama [1], José-P. García-Sabater [2], Francisco-Cruz Lario[3]

[1] Centro de Investigación Gestión e Ingeniería de la Producción (CIGIP). UPV. Camino de Vera s/n, Edifício 8G - Ingreso D - Nivel 1. Valencia. jocalla@doctor.upv.es y Departamento de Ingeniería Industrial de la Facultad de Ingeniería de la Universidad de Piura. Av. Ramón Mugica 131 – Piura – Perú. jcaldero@udep.edu.pe
[2] ROGLE. Departamento de Organización de Empresas Universidad Politécnica de Valencia. Camino de Vera s/n, 46022 Valencia. España. jpgarcia@omp.upv.es
[3] Centro de Investigación Gestión e Ingeniería de la Producción (CIGIP). UPV. Camino de Vera s/n, Edifício 8G - Ingreso D - Nivel 1. Valencia. fclario@omp.upv.es

Resumen

En este documento se presenta un modelo matemático determinista multiproducto para la planificación táctica de la producción de artículos de innovación (como son los productos de moda de una sola temporada) con procesos establecidos (con tecnología conocida), con dos o más alternativas de aprovisionamiento y dos o más niveles de materiales (lista de materiales). El modelo incluye la posibilidad de aplazar los procesos de subensamble y de montaje final, además considera varios medios y tiempos de transporte, capacidad finita en cada recurso de producción, almacén y transporte, y limitación del presupuesto para la compra de materiales.

Palabras clave: Planificación Táctica, Evaluación de Proveedores, Cadena de Suministro

1. Introducción

Actualmente, las cadenas de suministro abarcan empresas en distintos países, que planifican y coordinan sus capacidades (de producción, almacenamiento, transporte, etc.) para brindar al mercado productos en el tiempo, cantidad, variedad y coste adecuados. Por ejemplo la industria de la confección tiene el diseño, producción de telas, corte, costura y venta al por menor en diferentes partes del mundo (Fisher et al.

[1] El presente trabajo se enmarca dentro de las investigaciones realizadas conjuntamente entre el Departamento de Organización de Empresas y el Instituto Tecnológico de Informática de la Universidad Politécnica de Valencia dentro del proyecto Nuevos Algoritmos Bio-Inspirados en Logística Avanzada, presentado al Programa de Cooperación Tecnológica entre Centros de Investigación y Tecnología subvencionado por el Instituto para la Pequeña y Mediana Empresa de la Generalitat Valenciana.

1994). Muchas otras industrias, tales como equipo de telecomunicaciones y ordenadores, electrónica de consumo y juguetes, tienen una estructura de cadena de suministro semejante debido a los bajos costes de mano de obra de países del sudeste asiático, China, Centro América, etc. (Christopher et al., 2006). Esto obliga a planificar la producción abarcando dos o más niveles (montaje final, submontaje, fabricación de componentes, etc.) en dos o más empresas ubicadas en distintos países.

El presente documento analiza la planificación de la producción de artículos de innovación (como los productos de moda de una sola temporada) con procesos establecidos (con tecnología madura), demanda concentrada en dos o tres meses del año (aunque el resto del año hay una pequeña demanda) y dos o más alternativas de aprovisionamiento, un proveedor local (en el mismo país) y un proveedor lejano (en otro continente) por lo menos. Además considera la posibilidad de aplazamiento (*postponement*) tanto en las etapas de proceso del proveedor lejano como en el propio país.

Leung y Ng (2007b) han analizado el problema de un proveedor en un país lejano y muestran que en todos los casos (diferentes escenarios y distintos costes de escasez) es preferible el plan de producción con aplazamiento al plan de producción sin él, y que el ahorro es mayor cuanto mayor es la demanda y cuanto mayor es el coste de escasez.

En Calderón et al. (2008) se presenta un estado del arte de la planificación de la producción para la contratación de producción a proveedores industriales en una Cadena de Suministro. Arntzen et al. (1995), Lee et al. (2002), Chern y Hsieh (2007), Barbarosoglu (2000), Leung y Ng (2007a) y Peidro et al. (2007) han tratado el problema abarcando distintas partes del mismo y contribuyendo con diferentes enfoques. El presente trabajo se basa en los aportes de estos autores.

El resto del trabajo se organiza como sigue, en el segundo apartado se plantea el problema. En el tercer apartado se presenta el modelo desarrollado. En el cuarto se muestran las condiciones de los contratos y cómo se satisfacen. Las conclusiones se incluyen en el quinto apartado de este estudio.

2. Planteamiento del Problema

Este estudio abarca el caso general de Cadenas de Suministro integradas por empresas que diseñan productos de temporada y subcontratan la fabricación a distintos proveedores tanto nacionales como extranjeros (en países cuyo coste de producción es muy bajo). El número y localización de los proveedores es conocido. Los productos tienen demanda estacional con un pico alto en sólo dos o tres meses al año, luego del cual los productos pierden valor de mercado (teniendo un precio de remate muy bajo que puede estar incluso debajo del coste). El precio de los productos terminados es conocido y fijo hasta que acaba el pico de demanda.

Por lo anterior, la empresa debe planificar la producción de i productos en función de previsiones de demanda de los productos finales (sólo éstos tienen demanda externa), considerando dos o más niveles de componentes (lista de materiales) que pueden ser compartidos o no entre los diferentes productos, abarcando todo un año y actualizando el plan cada periodo (horizonte rodante truncado al periodo T) para subcontratar la producción “en firme” de los siguientes dos o tres periodos e “indicativa” de allí en adelante (considerando además que el tiempo de envío de los proveedores extranjeros

es mayor a un periodo). También se considera que la capacidad de los proveedores es finita pero distinta para cada periodo de tiempo y que el coste de las materias primas es diferente para cada proveedor. En el caso del proveedor local se supone un plazo de entrega menor o igual a un periodo (una semana).

El problema en sí consiste en decidir las cantidades de los distintos ítems (productos terminados –pt-, productos intermedios y componentes) a subcontratar a los proveedores en cada periodo de tiempo en función de sus tiempos de producción y transporte, los medios y costes de transporte, y los costes de producción, inventario y entregas diferidas (sólo los pt tienen entregas diferidas). Todos los costes pueden ser diferentes en los distintos periodos de tiempo y, especialmente en el caso de las entregas diferidas, el coste puede ir aumentando a medida que se aproxime al final del horizonte de planificación. El problema incluye además la posibilidad de aplazar el montaje parcial y el ensamble final de los productos en función de los costes de producción, transporte y almacenaje (de los componentes, productos intermedios y pt), y la posibilidad de hacer envíos (de los proveedores extranjeros a los proveedores nacionales) de componentes, productos intermedios y pt, para su almacenamiento y ensamble final local. Esto quiere decir que cada proveedor puede fabricar los productos finales de dos maneras: directamente de materias primas a producto terminado (productos directos) o producir componentes y almacenarlos, luego fabricar productos intermedios y almacenarlos, y finalmente ensamblar pt (estos últimos son llamados productos indirectos). En la nomenclatura del modelo propuesto se asigna los primeros F números "i" a los productos terminados directos y del F+1 al 2F a los indirectos.

3. Modelo propuesto

El presente modelo se basa en el trabajo de Peidro et al. (2007) pero se le incorpora estructuras de materiales alternativas y se le añade la posibilidad de hacer aplazamiento; luego se ejecuta con horizonte rodante de manera que se satisfagan los requerimientos de los proveedores (agregando una restricción).

El modelo permite determinar:

- Plan de producción de cada proveedor.
- Plan de transporte entre nodos (proveedores extranjeros y nacionales).
- Cantidad de ventas y beneficio final.
- Nivel de inventario de cada nodo.
- Todos los costes.

3.1. Nomenclatura

3.1.1. Índices / Conjunto

Índices / Conjunto	Significado
$t \in T$	Periodos ($t = 1, 2,\ldots, T$)
$i \in I$	Productos (materias primas, productos intermedios y productos terminados directos e indirectos). Si se fabrican F pt directos $i = 1, 2,\ldots, F, F+1,\ldots, I$

$n \in N$	Nodos de la CS ($n = 1, 2,\ldots, N$)
$j \in J$	Recursos de producción ($j = 1, 2,\ldots, J$)
$l \in L$	Modos de Transporte ($l = 1, 2,\ldots, L$)
$ip \in IP$	Productos de nivel inmediato superior en la lista de materiales ($ip = 1, 2,\ldots, IP$)
$id \in ID$	Productos directos ($id = 1, 2,\ldots, ID$)
$no \in NO$	Nodos de origen para transportes ($no = 1, 2,\ldots, NO$)
$nd \in ND$	Nodos de destino para transportes ($nd = 1, 2,\ldots, ND$)

3.1.2. Costes

Costes	Significado
CPV_{injt}	Coste variable de producción por unidad del producto *i* en *j* en
CPE_{njt}	Coste de horas extra en el recurso *j* en *n* en *t*
CPO_{njt}	Coste de tiempo ocioso en el recurso *j* en *n* en *t*
CMP_{int}	Coste por unidad de la materia prima *i* en *n* en *t*
$CTN_{inondlt}$	Coste de transporte por unidad de *i* de *no* a *nd* por medio de *l* e periodo *t*
CIN_{int}	Coste por unidad de inventario de *i* en *n* en *t*
CED_{int}	Coste de entregas diferidas de *i* en *n* en *t*
PV_{int}	Precio de venta de cada producto terminado *i* en *n* en *t*

3.1.3. Otros datos

Otros datos	Significado
$LMI_{ip,in}$	Cantidad de *i* para producir una unidad de *ip* en *n*
$LMD_{id,in}$	Cantidad de *i* para producir una unidad de *id* en *n*
$CMAX_{nt}$	Monto máximo para compras del nodo *n* en el periodo *t*
D_{int}	Demanda del producto *i* en *n* en *t*
$MAXPN_{njt}$	Capacidad máxima de producción en horas normales en *j*, *n* y *t*

$MAXHE_{njt}$	Capacidad máxima de producción en horas extra en *j* en *n* en *t*
Inv_{in0}	Cantidad en inventario del producto *i* en *n* en *t=0*
$MINP_{injt}$	Mínima cantidad a producir de *i* en *j* en *n* en *t*
ED_{in0}	Entregas diferidas del producto *i* en *n* en *t=0*
$RTNI_{inondlt}$	Cantidades de *i* a recibir en *nd* proveniente de *no* por medio de *l* en los periodos *t*= 1, 2,…,*TTN*
$TCN_{inondl0}$	Cantidad en transporte de *no* a *nd* por medio de *l* en el periodo *t* = 0
TTN_{nondlt}	Tiempo de transporte de *no* a *nd* por medio de *l* en el periodo *t*
TP_{inj}	Tiempo de producción por unidad de *i* en *j* en *n*
V_i	Volumen físico del producto *i*
$VMAXL_{nt}$	Capacidad máxima de transporte de *l* en *t*
$VMAXI_{nt}$	Capacidad máxima de inventario en *n* en *t*

3.1.4. Variables binarias

Variables binarias	Significado
X^1_{nondlt}	Variable que toma el valor de 1 si $TTN_{nondlt} > 0$, y 0 para los demás casos
X^2_{nondlt}	Variable que toma el valor de 1 si $TTN_{nondlt} = 0$, y 0 para los demás casos

3.1.5. Variables de decisión

Variables de decisión	Significado
P_{injt}	Producción (cantidad) de producto *i* en *j* en *n* en *t*
S_{int}	Suministro (cantidad) de producto *i* en *j* en *n* en *t*
ED_{int}	Entregas diferidas de *i* en *n* al final del periodo *t*
$TNN_{inondlt}$	Cantidad de *i* despachada de *no* a *nd* por medio de *l* en el periodo *t con no* <> *nd y* $CTN_{inondlt} > 0$

$RTN_{inondlt}$	Cantidad de *i* recibida en *nd* proveniente de *no* por medio de *l* en el periodo *t con no <> nd y* $CTN_{inondlt} > 0$
$TCN_{inondlt}$	Cantidad de *i* en transporte de *no* a *nd* por medio de *l* en el periodo *t con no <> nd y* $CTN_{inondlt} > 0$
Inv_{int}	Cantidad en inventario del producto *i* en *n* al final del periodo *t*
CC_{int}	Cantidad comprada de *i* en *n* en *t*
Tex_{njt}	Horas extra trabajadas en el recurso *j* en *n* en *t*
Toc_{njt}	Horas ociosas en el recurso *j* en *n* en *t*
Y_{injt}	Variable binaria que indica si el producto i ha sido producido en el recurso *j* en *n* en *t*

3.2. Modelo Matemático

3.2.1. Objetivo

$$Max(Z) = \sum_{i}^{I}\sum_{n}^{N}\sum_{t}^{T} PV_{\text{int}} S_{\text{int}} - \sum_{i}^{I}\sum_{n}^{N}\sum_{j}^{J}\sum_{t}^{T} CPV_{injt} P_{injt} - \sum_{n}^{N}\sum_{j}^{J}\sum_{t}^{T} (CPE_{njt} Tex_{njt} + CPO_{njt} Toc_{njt})$$
$$- \sum_{i}^{I}\sum_{n}^{N}\sum_{t}^{T} (CMP_{\text{int}} CC_{\text{int}} + CIN_{\text{int}} Inv_{\text{int}} + CED_{\text{int}} ED_{\text{int}}) - \sum_{i}^{I}\sum_{no}^{NO}\sum_{nd}^{ND}\sum_{l}^{L}\sum_{t}^{T} CTN_{inondlt} TNN_{inondlt}$$

(1)

La función objetivo (1) incluye las ganancias en primer término y luego las sumatorias de todos los costes. El segundo término es la suma de los costes de producción. El tercer término suma los costes de horas extra y de tiempo ocioso. El cuarto término suma los costes de compra de materiales, inventario y entregas diferidas. El último término suma los costes de transporte. Al final del último período, el coste de entregas diferidas es equivalente al coste de escasez (que es el coste de ventas perdidas) y el coste de inventario es equivalente al coste de obsolescencia (productos que tendrán que ser rematados).

3.2.2. Restricciones

$$\sum_{i=1}^{I} TP_{inj} \left(\sum_{id}^{ID} P_{i=id,nJt} LMD_{id,in}\right) + \sum_{i=1}^{I} P_{injt} TP_{inj} \le MAXPN_{njt} + MAXHE_{njt}$$

$$\forall\ n, j, t \qquad \textbf{(2)}$$

La restricción (2) hace que el tiempo de producción real sea menor que la capacidad máxima en horas normales más horas extra en cada nodo, recurso de producción y periodo. El primer término de la inecuación es para los productos terminados directos; como éstos no generan inventarios de productos intermedios pero sí usan los recursos

de producción se debe usar la lista de materiales (LMD) para determinar su carga de trabajo en cada recurso.

$$(\sum_{id}^{ID} P_{i=id,nJt} LMD_{id,in}) TP_{inj} + P_{ijnt} TP_{inj} \leq MAXPN_{njt} Y_{injt} + MAXHE_{njt} Y_{injt}$$

$$\forall\ i, n, j, t \qquad \textbf{(3)}$$

$$(\sum_{id}^{ID} P_{i=id,nJt} LMD_{id,in}) + P_{ijnt} \geq MINP_{injt} Y_{injt} \quad \forall\ i, n, j, t \quad \textbf{(4)}$$

Las restricciones (3) a (4) aseguran que la producción se realice por encima del mínimo posible en los diferentes recursos. En estas tres primeras restricciones el primer término es para los descendientes de los productos directos y el segundo término es para todos los productos terminados, subensambles y componentes.

$$Inv_{\text{int}} = Inv_{in,t-1} + \sum_{j}^{J} P_{injt} + \sum_{no}^{NO} \sum_{l}^{L} RTN_{ino,nd=n,lt} + CC_{\text{int}} -$$
$$- \sum_{nd}^{ND} \sum_{l}^{L} TNN_{i,no=n,ndlt} - S_{\text{int}} - \sum_{ip=1}^{IP} (LMI_{ip,in} \sum_{j}^{J} P_{i=ip,njt}) \qquad \forall\ i, n, t$$

(5)

La restricción (5) calcula el inventario en cada nodo al final de cada periodo. El inventario para un artículo i es igual al inventario del periodo anterior más la cantidad producida del producto i en todos los recursos de ese mismo nodo, más las recepciones (producto i fabricado en otro nodo), más la cantidad comprada (sólo si i es materia prima), menos los despachos a otros nodos, menos las salidas por ventas (sólo para productos terminados) menos lo que se consume en el mismo nodo para fabricar un producto de nivel inmediato superior que lleve i (sólo para materias primas y productos intermedios). En esta restricción, cuando t = 1, el inventario inicial $Inv_{i,n,t-1}$ asume el valor del dato $Inv_{i,n,0}$.

Los productos terminados directos (i = 1, 2,…, F) no generan inventarios de productos intermedios y pasan directamente de materia prima a producto final; esto se consigue a través de su lista de materiales (LMI), la cual relaciona la materia prima con el producto terminado directo. Los productos terminados indirectos sí consumen subensambles, éstos consumen componentes y los componentes consumen materias primas. La lista de materiales (LMI) brinda la información para ambos tipos de productos.

$$RTN_{inondlt} = RTNI_{inondlt} + TNN_{inondl,t-TTN_{nondl}} \quad \forall\ i, no, nd, l, t \quad \textbf{(6)}$$

$$TCN_{inondlt} = TCN_{inondl,t-1} + TNN_{inondlt} - RTN_{inondlt} \quad \forall\ i, no, nd, l, t \quad \textbf{(7)}$$

Las restricciones (6) y (7) controlan el transporte de productos entre nodos. En la ecuación (6) la recepción de envíos es igual a las recepciones programadas ($RTNI_{inondlt}$) para los primeros TTN periodos (es decir envíos que se despacharon antes del periodo 1) más los despachos que se realicen a partir del periodo 1 (segundo término del lado

derecho de la ecuación). En la ecuación (7) las cantidades en tránsito entre nodos son iguales a la cantidad en transporte del periodo anterior más los despachos iniciados en el mismo período menos las recepciones en el mismo periodo. El $TCN_{inondl,t-1}$ asume el valor del dato $TCN_{inondl0}$ cuando t = 1. La restricción (8) impide que se supere la capacidad de almacenaje de los nodos en cualquier periodo.

$$\sum_{i}^{I} Inv_{\text{int}} V_i \leq VMAXI_{nt} \qquad \forall\, n, t \quad \textbf{(8)}$$

$$\sum_{i}^{I}\sum_{no}^{NO}\sum_{nd}^{ND} TCN_{inondlt} V_i \chi^1_{nondlt} + \sum_{i}^{I}\sum_{no}^{NO}\sum_{nd}^{ND} TNN_{inondlt} V_i \chi^2_{nondlt} \leq VMAXL_{lt} \quad \forall\, l, t$$

(9)

La ecuación (9) limita la cantidad a ser transportada a través de un determinado medio. Suma las cantidades en tránsito más los despachos iniciados en el periodo. Dado que para los envíos con *TTN* igual a cero el *TCN* (productos en tránsito) es cero, se debe considerar en el segundo término de la ecuación los traslados que se realizan dentro del mismo período. Además, el término a la derecha de la desigualdad puede variar de un período a otro.

$$\sum_{i}^{I} CC_{\text{int}} CMP_{\text{int}} \leq CMAX_{nt} \qquad \forall\ n, t \quad \textbf{(10)}$$

Esta restricción (10) limita la cantidad a comprar de cada nodo en cada periodo. El lado derecho de la desigualdad establece el monto máximo disponible para compra (en unidades monetarias). Esta es una restricción que depende del presupuesto disponible.

$$ED_{\text{int}} = ED_{in,t-1} + D_{\text{int}} - S_{\text{int}} - S_{i+F,nt} \qquad \text{para } i=1,\ldots,F,\ \forall\ n, t \quad \textbf{(11)}$$

La restricción (11) es sólo para productos terminados, de los cuales hay tanto directos como indirectos. Por cada artículo hay un i directo y un i+F indirecto. Si la suma de ambos no es igual a la demanda más los productos pendientes del periodo anterior, entonces habrá pedidos pendientes al final del periodo. Cuando t=1 el $ED_{in,t-1}$ asume el valor del dato ED_{in0}. En el último periodo, el ED_{inT} será el número de productos que faltaron para satisfacer la demanda en esa temporada y por lo tanto, para minimizar su número, se debe poner un coste elevado CDE_{inT}, así mismo el CIN_{inT} debe ser elevado para disminuir la obsolescencia (número de artículos que sobran al final de la temporada).

$$Tex_{njt} = \sum_{i=1}^{I} TP_{inj}\left(\sum_{id}^{ID} P_{i=id,nJt} LMD_{id,in}\right) + \sum_{i=1}^{I} P_{injt} TP_{inj} - MAXPN_{njt} + Toc_{njt} \quad \forall\ \ n, j, t \quad \textbf{(12)}$$

La ecuación (12) determina si hay horas extra o tiempo ocioso en cada recurso, nodo y periodo. Si la producción del periodo (dos primeros sumandos del lado derecho) es mayor que la capacidad máxima en horas normales (tercer término del lado derecho) entonces *Tex* tendrá valor positivo y *Toc* será cero. Si la producción es igual que la capacidad tanto *Tex* como *Toc* serán cero y si la producción es menor que la capacidad, *Toc* será positivo y *Tex* será cero.

$$P_{injt} \geq 0 \qquad \forall\ i, n, j, t \quad \textbf{(13)}$$

$$S_{int,}\, ED_{int,}\, Inv_{int,}\, CC_{int} \geq 0 \qquad \forall\ i, n, t \quad \textbf{(14)}$$

$$RTN_{inondlt},\, TCN_{inondlt},\, TNN_{inondlt} \geq 0 \qquad \forall\ i, no, nd, l, t \quad \textbf{(15)}$$

$$Tex_{njt}, Toc_{njt} \geq 0 \qquad \forall\ n, j, t \quad \textbf{(16)}$$

Las restricciones (13) a (16) son de no negatividad para las variables de decisión.

4. Condiciones de los contratos

En el apartado dos se indica que los proveedores requieren que se subcontrate la producción "en firme" con cierta anticipación (varios periodos para los proveedores extranjeros y pocos periodos para los proveedores nacionales) y de forma "indicativa" para periodos posteriores. A esta anticipación mínima se le llama "plazo de entrega" (PE) y es diferente para cada proveedor. Además, los proveedores requieren que la diferencia entre la cantidad subcontratada en un periodo y la del siguiente no sea muy elevada y por lo tanto fijan un límite a dicha variación (β); esta diferencia puede aumentar (fijando valores de β mayores) a medida que los dos periodos considerados están más lejos en el futuro.

Si en cada periodo *t* se compromete "en firme" la producción del periodo *t*+PE, se denomina $COM_t(k)$ al compromiso hecho en el periodo k para ser entregado en el periodo *t* para *t*=*k*+PE. Entonces, cada nuevo pedido comprometido (PC_t) debe satisfacer:

$$(1-\beta_t)\, COM_t(k) \leq PC_t \leq (1+\beta_t)\, COM_t(k) \qquad \text{para } 0 \leq \beta_t \leq 1, \text{ para } t=k+\text{PE} \quad \textbf{(17)}$$

Para lograr lo anterior es necesario que el modelo se ejecute bajo el esquema de horizonte rodante considerando los primeros PE periodos como pedidos comprometidos en todos los proveedores. En el modelo se asume que los proveedores extranjeros pueden enviar (TNN) sus componentes, productos intermedios y pt a los proveedores locales y que sólo los proveedores locales tienen demanda externa, así en la restricción (5) de inventario sólo los proveedores locales de pt usarán el término S_{int}. Con el enfoque rodante, el modelo propuesto se ejecuta T veces (T = número de periodos del horizonte de planificación) y se agrega un índice *u* a todas las variables y parámetros para identificar sus valores en cada ejecución (*u*=1,…,T). La primera ejecución del modelo, EJE(1), se resuelve considerando dados (como datos) los valores de los envíos (*TNN*) y los suministros (S_{int}) de los primeros PE periodos; para generalizar se llama $ES_t(u)$ a estas variables de decisión de manera que si, por ejemplo, asumimos que el PE es de 3 períodos, entonces para la primera ejecución $ES_1(1)$, $ES_2(1)$ y $ES_3(1)$ son los valores comprometidos iniciales.

Luego el modelo es "rodado" cambiando el inicio al periodo t=2 y los pedidos comprometidos para los nuevos tres primeros periodos toman los valores ES1(2) = ES2(1), ES2(2) = ES3(1) y ES3(2) = PC4(1). El PC4(1) es el *TNN* de los proveedores extranjeros y el S_{int} de los proveedores locales del periodo 4 de la primera ejecución del modelo. Así, el PCt del primer periodo después del PE se convierte en pedido comprometido de la siguiente ejecución. Se incluye la ecuación (17) en la segunda ejecución, EJE(2), y los pedidos flexibles para los siguientes periodos asumen el valor COMt(2) = PCt+1(1) para t > 3 (bajo el supuesto de PE=3). Esto se repite para todo el horizonte de planificación "rodando" los pedidos comprometidos y flexibles; al final los pedidos del cuarto periodo en cada ejecución se consideran como los pedidos más realistas a los proveedores (Barbarosoglu; 2000). Este procedimiento se sistematiza de

la siguiente manera:

I. Hacer u=1 y ejecutar el modelo EJE(1) usando los compromisos $ES_t(u)$ de los PE primeros periodos para determinar $PC_{PE+1}(u)$.

II. "Rodar" el primer periodo y poner como tiempo de inicio el siguiente periodo, y hacer u=u+1.

III. Actualizar los inventarios iniciales.

IV. Actualizar los pedidos comprometidos: $ES_1(u)=ES_2(u\text{-}1)$, $ES_2(u)=ES_3(u\text{-}1)$,…, $ES_{PE}(u)=PC_{PE+1}(u\text{-}1)$.

V. Actualizar los pedidos flexibles: $COM_t(u) = PC_{t+1}(u\text{-}1)$ para t = PE, …, T-u+1.

VI. Agregar la restricción (17) incluyendo $COM_t(u)$.

VII. Resolver EJE(u).

VIII. Repetir los pasos II a VII hasta el final del horizonte de planificación.

5. Conclusiones

El modelo propuesto permite obtener soluciones óptimas y proporciona las cantidades de productos terminados a fabricar en forma directa y en forma indirecta (con aplazamiento), asimismo las cantidades de productos intermedios y componentes a producir y transportar, y cuándo hacer el ensamble final (tanto en el proveedor lejano como en el nacional). Gracias a su estructura, se puede tener componentes compartidos por los distintos productos finales y se facilita su transporte y control de inventarios.

Mediante el enfoque de horizonte rodante se puede planificar los pedidos a los proveedores de manera que se cumpla con sus requerimientos de plazo de entrega y de cambio en el tamaño de los pedidos. Esto permitirá contratar capacidades con adecuada anticipación.

Además, a partir de cada solución se puede hallar el nivel de servicio de cada producto terminado y el general, y compararlo con las otras alternativas. También se puede hallar el beneficio por producto de manera que se pueda decidir de cuál conviene producir mayor o menor cantidad (por esto es mejor que un modelo que sólo minimiza costes).

El modelo se ha implementado en MPL-CPLEX y para su ejecución con horizonte rodante se ha desarrollado un algoritmo en Delphi, el cual además permite hacer análisis de sensibilidad para determinar en qué condiciones compensará contratar la producción a proveedores cercanos frente a proveedores lejanos o a ambos en distintos periodos y cantidades.

Este modelo se puede ampliar para considerar economías de escala en los costes de transporte y compra de materiales, tamaño mínimo y máximo del lote de producción, contratación y despido de la mano de obra directa, etc. El modelo es determinista pero se puede modificar para considerar la demanda y otros parámetros con incertidumbre. Se puede seguir el enfoque de Peidro et al. (2007) mediante lógica difusa (*fuzzy*) o el de Leung y Ng (2007b) mediante programación estocástica de dos etapas con cambio de plan y análisis de escenarios.

Referencias

Arntzen, B.C.; Brown, G.G.; Harrison, T.P.; Trafton, L.L. (1995). Global Supply Chain Management at Digital Equipment Corporation. Interfaces, Vol. 25, No. 1, pp. 69-93.

Barbarosoglu, G, (2000). An integrated supplier-buyer model for improving supply chain coordination. Production Planning & Control, Vol. 11, Nº 8, pp. 732–741.

Calderón-Lama, J-L.; García-Sabater, J-P.; Lario, F-C. (2008). Estado del arte de la planificación de la producción para la contratación de producción a proveedores industriales en una Cadena de Suministro. XII Congreso de Ingeniería de Organización. 2nd International Conference on Industrial Engineering and Industrial Management. Burgos, Septiembre 3 al 5.

Chern, C.-C.; Hsieh J.-S. (2007). A heuristic algorithm for master planning that satisfies multiple objectives. Computers & Operations Research, Vol. 34, No. 11, pp. 3491–3513.

Christopher, M.; Peck, H.; Towill, D. (2006). A taxonomy for selecting global supply chain strategies. The International Journal of Logistics Management, Vol. 17, Nº2, pp. 277-287.

Fisher, M.; Hammond, J.; Obermeyer, W.; Raman, A. (1994). Making supply meet demand in an uncertain world. Harvard Business Review, Vol. 72, Nº 3, pp. 83-93.

Lee, Y.H.; Kim, S.H.; Moon, Ch. (2002). Production-distribution planning in supply chain using a hybrid approach. Production Planning & Control, Vol. 13, No. 1, pp. 35–46.

Leung, S.C.H.; Ng, W-L. (2007a). A goal programming model for production planning of perishable products with postponement. Computers & Industrial Engineering, Vol. 53, No. 3, pp. 531-541.

Leung, S.C.H.; Ng, W-L. (2007b). A stochastic programming model for production planning of perishable products with postponement. Production Planning & Control, Vol. 18, No. 3, pp. 190-202.

Peidro, D.; Mula, J.; Poler, R. (2007). Supply chain planning under uncertainty: a fuzzy linear programming approach. Fuzzy Systems Conference. IEEE International. Vol. 1, pp. 1-6.

Anexo 2

Herramienta de implementación: algoritmos

Clase para crear XML de entrada

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.util.Random;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.datatype.DatatypeConfigurationException;
import
com.sun.xml.internal.bind.v2.runtime.RuntimeUtil.ToStringAdapter;
import LectorXML.ActivaCompra;
import LectorXML.BOM;
import LectorXML.CoefTecno;
import LectorXML.CosCapacidad;
import LectorXML.CosteBackOrder;
import LectorXML.CosteInventory;
import LectorXML.CosteStroke;
import LectorXML.Data;
import LectorXML.DemBass;
import LectorXML.DemandaEs;
import LectorXML.DemandaRe;
import LectorXML.InventarioIncial;
import LectorXML.Leadtime;
import LectorXML.ObjectFactory;
import LectorXML.PenalidadMAS;
import LectorXML.PenalidadMENOS;
import LectorXML.PrecioCompra;
import LectorXML.PrecioProd;
import LectorXML.ProbabilidadEsc;
import LectorXML.RBOM;
import LectorXML.ResidualInv;
import LectorXML.UpperBounCapacidad;

public class CrearXML {

public CrearXML() {
         super();}

//Fin constructor de la clase

public static void crearxmlPres(String fsalida, LeerExcel le) throws
DatatypeConfigurationException,DatatypeConfigurationException,
JAXBException, FileNotFoundException{

//Creación de un xml

JAXBContext jaxbContext = JAXBContext.newInstance("LectorXML");

//Nombre del paquete

                  Marshaller marshaller =
jaxbContext.createMarshaller();
```

```
marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,new
Boolean(true));

//Factoría de objetos del esquema del que se tomara
posteriormente la raiz

            ObjectFactory of = new ObjectFactory();

            //Se crea un objeto que va a ser la carpeta raiz
Data datos =of.createData();//Se crea el objeto del esquema of

   //Definimos el número de elementos de cada conjunto de
parámetros

datos.setNumI(le.numI); //se le da valor al atributo numI
datos.setNumE(le.numE); //se le da valor al atributo numI
datos.setNumO(le.numO); //se le da valor al atributo numI
datos.setNumK(le.numK); //se le da valor al atributo numI
datos.setNumT(le.numT); //se le da valor al atributo numI
datos.setNumR(le.numR); //se le da valor al atributo numI

/* Se crean cada uno de las subcarpetas que representaran a
 * cada uno de los parámetros del problema del transporte
 */

datos.setTao(le.tao);
datos.setIniciaDemanda(le.inidem);

for(int i=0; i<le.numI;i++){
   datos.getConjuntoSKU().add(le.getSKU[i]);}

for(int t=0; t<le.numT;t++){
datos.getConjuntoPeriodos().add(Integer.toString(le.getPeriodos
[t]));}

for(int o=0; o<le.numO;o++){

datos.getConjuntoEventos().add(Integer.toString(le.getEventos[o
]));  }

   //System.out.println("el valor del ocurrencia es
"+le.getEventos[o]);

for(int e=0; e<le.numE;e++){

datos.getConjuntoEscenarios().add(Integer.toString(le.getEscena
rios[e]));}

for(int r=0; r<le.numR;r++){
   datos.getConjuntoRecursos().add(le.getRecursos[r]);}

for(int k=0; k<le.numK;k++){

datos.getCojuntoStrokes().add(le.getStroke[k]);}

   //Conjunto de parámetros del modelo matemático

ActivaCompra AC =of.createActivaCompra(); //1
PrecioProd pp =of.createPrecioProd(); //2
PrecioCompra pc = of.createPrecioCompra();//3
ResidualInv resi= of.createResidualInv();//4
```

```
CosteBackOrder bak=of.createCosteBackOrder();//5
CosteInventory ci=of.createCosteInventory();//6
InventarioIncial inis=of.createInventarioIncial();//7
DemandaEs dem =of.createDemandaEs(); //8
DemandaRe dr = of.createDemandaRe();//9
CosteStroke ck =of.createCosteStroke();//10
Leadtime lt =of.createLeadtime();//11
CosCapacidad ccap=of.createCosCapacidad();//12
UpperBounCapacidad up =of.createUpperBounCapacidad();//13
PenalidadMAS pms=of.createPenalidadMAS();//14
PenalidadMENOS pmm=of.createPenalidadMENOS();//15
BOM mbom=of.createBOM();//16
RBOM nbom =of.createRBOM();//17
ProbabilidadEsc Prob =of.createProbabilidadEsc();
CoefTecno Re =of.createCoefTecno();

for(int e=0; e<le.numE;e++){
   Prob.setE(Integer.toString(le.getEscenarios[e]));
   Prob.setPe(le.pe[e]);
   datos.getProbabilidades().add(Prob);
   Prob=of.createProbabilidadEsc();}

//Escribir demanda estocástica
for(int i=0;i<le.numI;i++){
for(int t=0;t<le.numT;t++){
for(int e=0;e<le.numE;e++){
                    dem.setI(le.getSKU[i]);
            dem.setT(Integer.toString(le.getPeriodos[t]));
            dem.setE(Integer.toString(le.getEscenarios[e]));
dem.setDESite(le.dEMite[i][t][e]);
datos.getDemEsS().add(dem);
dem=of.createDemandaEs();}}}

//Escribir demanda real por eventos
for(int i=0;i<le.numI;i++){
   for(int t=0;t<le.numT;t++){
   for(int o=0;o<le.numO;o++){
            dr.setI(le.getSKU[i]);
            dr.setT(Integer.toString(le.getPeriodos[t]));
            dr.setO(Integer.toString(le.getEventos[o]));
            dr.setDito(le.dEMito[i][t][o]);
            datos.getDemRealS().add(dr);
            dr=of.createDemandaRe();}}}

//Precio de venta de producto
for(int i=0; i<le.numI;i++){
            pp.setI(le.getSKU[i]);
            pp.setPVi(le.pVi[i]);
            datos.getPrecios().add(pp);
            pp=of.createPrecioProd();
            inis.setI(le.getSKU[i]);
            inis.setINI(le.iNI[i]);
            datos.getInvInicial().add(inis);
            inis=of.createInventarioIncial();

//Compras
for(int t=0; t<le.numT;t++){
                    AC.setI(le.getSKU[i]);
            AC.setT(Integer.toString(le.getPeriodos[t]));
            AC.setACit(le.aCit[i][t]);
            datos.getCompras().add(AC);
```

```
            AC=of.createActivaCompra();

//Precio de compra
            pc.setI(le.getSKU[i]);
            pc.setT(Integer.toString(le.getPeriodos[t]));
            pc.setPCit(le.pCit[i][t]);
            datos.getPrecioCompras().add(pc);
            pc=of.createPrecioCompra();

//Todos los que sea i,t
            resi.setI(le.getSKU[i]);
            resi.setT(Integer.toString(le.getPeriodos[t]));
            resi.setVRit(le.vRit[i][t]);
            datos.getValorResidualI().add(resi);
            resi=of.createResidualInv();

            bak.setI(le.getSKU[i]);
            bak.setT(Integer.toString(le.getPeriodos[t]));
            bak.setCBit(le.cBit[i][t]);
            datos.getCosBackorders().add(bak);
            bak=of.createCosteBackOrder();

            ci.setI(le.getSKU[i]);
            ci.setT(Integer.toString(le.getPeriodos[t]));
            ci.setCH(le.cHit[i][t]);
            datos.getCosteAlmacena().add(ci);
            ci=of.createCosteInventory();       }
}

   for(int k=0;k<le.numK;k++){
            lt.setK(le.getStroke[k]);
            lt.setLTk(le.lt[k]);
            datos.getLeadtimes().add(lt);
            lt=of.createLeadtime();

   for(int t=0;t<le.numT;t++){
            ck.setK(le.getStroke[k]);
            ck.setT(Integer.toString(le.getPeriodos[t]));
            ck.setCOkt(le.cOkt[k][t]);
            datos.getCosStrokes().add(ck);
            ck=of.createCosteStroke();}

   for(int r=0;r<le.numR;r++){
            Re.setK(le.getStroke[k]);
            Re.setR(le.getRecursos[r]);
            Re.setRErk(le.rEkr[k][r]);
            datos.getCoeRecurs().add(Re);
            Re=of.createCoefTecno();}
            }

   for(int r=0;r<le.numR;r++){
   for(int t=0;t<le.numT;t++){
            ccap.setR(le.getRecursos[r]);
            ccap.setT(Integer.toString(le.getPeriodos[t]));
            ccap.setCCAPrt(le.cCAPrt[r][t]);
            datos.getCostesCapacidad().add(ccap);
            ccap=of.createCosCapacidad();

            up.setR(le.getRecursos[r]);
            up.setT(Integer.toString(le.getPeriodos[t]));
            up.setKAPrt(le.kAPrt[r][t]);
```

```
                datos.getLimiteRecursos().add(up);
                up=of.createUpperBounCapacidad();

                pms.setR(le.getRecursos[r]);
                pms.setT(Integer.toString(le.getPeriodos[t]));
                pms.setPNmasrt(le.pEN1[r][t]);
                datos.getPenalidadesMas().add(pms);
                pms=of.createPenalidadMAS();

                pmm.setR(le.getRecursos[r]);
                pmm.setT(Integer.toString(le.getPeriodos[t]));
                pmm.setPNmenosrt(le.pEN2[r][t]);
                datos.getPenalidadesMenos().add(pmm);
                pmm=of.createPenalidadMENOS();}
                }

    for(int k=0;k<le.numK;k++){
    for(int i=0;i<le.numI;i++){
                mbom.setI(le.getSKU[i]);
                mbom.setK(le.getStroke[k]);
                mbom.setMik(le.mik[i][k]);
                datos.getBOMs().add(mbom);
                mbom=of.createBOM();

                nbom.setI(le.getSKU[i]);
                nbom.setK(le.getStroke[k]);
                nbom.setNik(le.nik[i][k]);
                datos.getRBOMs().add(nbom);
                nbom=of.createRBOM();}
                }

                marshaller.marshal(datos,new
FileOutputStream(fsalida));
                System.out.println("Creo XML:"+fsalida);}
        }
```

Lectura de la Hoja de Cálculo

```
import java.util.*;
import java.io.File;
import java.io.IOException;
import jxl.Cell;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;

/**
* En esta clase se leen los datos tomados de XML que guardados
en variables para su posterior tratamiento
 */

public class LeerExcel {

   // Declaración de atributos del objeto: problema a resolver
   public static int inidem;

//Número de elementos de cada conjunto
   public static int numI;
   public static int numT;
   public static int numK;
            public static int numR;
```

```
   public static int numE;
   public static int tao;
   public static int numO;

   //Parámetros del modelo de difusión
   public static double[] pAlfa;
   public static double[] pBeta;
   public static double[] qAlfa;
   public static double[] qBeta;
   public static double[] m1;
   public static double[] m2;
   public static double[] m3;
   public static double[] errori;

   //Elementos que forman los conjuntos
   public static String []getSKU ;
   public static int[]getEscenarios ;
   public static String []getStroke ;
   public static String []getRecursos;
   public static int[]getPeriodos;
   public static int[] getEventos;

            /**
 * Este parámetro es la demanda del SKU i en el tiempo t
generada para el escenario e
 */
public static double [][][] dEMite;//Demanda del SKU i en t en
el escenario e

/**
 * Este parámetro es la demanda real del  SKU i en el tiempo t
 */

public static double[][][] dEMito;//Demandas generadas por
ocurrencias
public static double [][] dEMit;
// colocar la otra dimencion a la Variable dEMit la de las
ocurrencias

/**
 * Este parámetro es el Precio de venta del producto i
 */
public static double [] pVi;

/**
 * Este parametro es el Valor Residual del inventario del SKU i
en el periodo t
 */
public static double [][] vRit;

   /**
    * Este parámetro es el precio al cual se compra la materia
prima y los componentes
    */
   public static double [][] pCit;

   /**
    * Este parámetro se refiere al coste de un STROKE k en el
periodo t
    */
   public static double [][] cOkt;
```

```
/**
 * Parámetro del coste de reserva de capacidad del recurso r en
el periodo t
 */
public static double [][] cCAPrt;

/**
 * Matriz de componentes i requeridos para un stroke k
 */
public static double [][] mik;

/**
 * Matriz de componentes i generados por un stroke k
 */
public static double [][] nik;

/**
 * Coeficientes Tecnológicos del consumo del recurso por parte
del stroke k
 */
public static double [][] rEkr;

/**
 * Activación de SKU a comprar en dimensiones i e t
 */
public static double [][] aCit;

/**
 * Capacidad máxima ofrecida por un proveedor de capacidad de
recursos
 */
public static double [][] kAPrt;

/**
 * Probabilidad de ocurrencia del escenario e
 */
public static double [] pe;
public static double [] iNI;//Inventario Inicial
public static int La;//Periodo Inicial
public static int Lb;//Periodo Final
public static int []lt; //Lead Time del stroke K
public static double [][] pEN1; //Penalidad
public static double [][] pEN2; //Penalidad
public static double[][] cHit; // Coste de Almacenamiento
public static double [][] cBit; // Coste de REtrasos

/**
 * Este método lee un XML de entrada para inicializar el
problema
 * @param sEntrada
 * Las entradas son un XML con la informacion de un problema
especifico a resolver
             */

   public LeerExcel(){

   super();
   }

   public static void LeerExcel(String problem){
```

```
   Workbook libro;

  try {
  libro = Workbook.getWorkbook(new File(problem));

      Sheet hoja1 = libro.getSheet(0);
      Sheet hoja2 = libro.getSheet(1);
      Sheet hoja3 = libro.getSheet(2);
      Sheet hoja4 = libro.getSheet(3);
      Sheet hoja5 = libro.getSheet(4);
      Sheet hoja6 = libro.getSheet(5);
      Sheet hoja7 = libro.getSheet(6);
      Sheet hoja8 = libro.getSheet(7);
      Sheet hoja9 = libro.getSheet(8);
      Sheet hoja10 = libro.getSheet(9);
      Sheet hoja11 = libro.getSheet(10);
      Sheet hoja12 = libro.getSheet(11);
      Sheet hoja13 = libro.getSheet(12);
      Sheet hoja14 = libro.getSheet(13);
      Sheet hoja15 = libro.getSheet(14);
      Sheet hoja16 =libro.getSheet(15);

Cell a1 =hoja1.getCell(1,1);
      String pasaa1 = a1.getContents();
           numI=Integer.parseInt(pasaa1);

Cell a2 =hoja1.getCell(1,2);
      pasaa1 = a2.getContents();
           numE=Integer.parseInt(pasaa1);

           a2 =hoja1.getCell(1,3);
      pasaa1 = a2.getContents();
           numK=Integer.parseInt(pasaa1);

           a2 =hoja1.getCell(1,4);
      pasaa1 = a2.getContents();
           numR=Integer.parseInt(pasaa1);

           a2 =hoja1.getCell(1,5);
      pasaa1 = a2.getContents();
           numT=Integer.parseInt(pasaa1);

           a2 =hoja1.getCell(8,0);
      pasaa1 = a2.getContents();
           numO=Integer.parseInt(pasaa1);

           a2 =hoja1.getCell(8,6);
      pasaa1 = a2.getContents();
           tao=Integer.parseInt(pasaa1);

           a2 =hoja1.getCell(8,8);
      pasaa1 = a2.getContents();
           inidem=Integer.parseInt(pasaa1);

           getSKU=new String[numI];
           for(int i=0; i<numI;i++){
                    a2 =hoja1.getCell(3,1+i);
               getSKU[i] = a2.getContents();

           }
```

```
                getStroke=new String[numK];
                for(int k=0; k<numK;k++){
                        a2 =hoja1.getCell(4,1+k);
                    getStroke[k] = a2.getContents();

                }

                getRecursos=new String[numR];
                for(int r=0; r<numR;r++){
                        a2 =hoja1.getCell(5,1+r);
                    getRecursos[r] = a2.getContents();

                }

        getPeriodos=new int [numT];
                for(int t=0; t<numT;t++){
                        getPeriodos[t] = t+1;
                }

                getEventos=new int [numO];
                for(int o=0; o<numO;o++){
                        getEventos[o] = o+1;
                }

                getEscenarios=new int [numE];
                pe=new double[numE];
                for(int e=0; e<numE;e++){

                getEscenarios[e] = (e+1); //Conjunto de escenaris

                a2 =hoja1.getCell(1,9+e);
                pasaa1 = (a2.getContents()).replace(',', '.');
                pe[e]=Double.parseDouble(pasaa1);
                }

                Cell a3=hoja2.getCell(4,0);
                pasaa1 =a3.getContents();
                int contara =Integer.parseInt(pasaa1);

                Cell a;

                dEMite=new double[numI][numT][numE];

                int contis=0;
                for(int e=0;e<numE;e++){
                for(int i=0;i<numI;i++){
                for(int t=0;t<numT;t++){

                a=hoja2.getCell(3, 1+contis+t);
                pasaa1 =a.getContents().replace(',', '.');


        dEMite[i][t][e]=Double.parseDouble(pasaa1);

                if (t==(numT-1)){
                contis=t+contis+1;
                        }//Fin If
                        }
                }
                }
```

```
            m1=new double[numI];
            for(int i=0;i<numI;i++){
            a=hoja3.getCell(1,2+i);
            pasaa1 =a.getContents().replace(',', '.');
            m1[i]=Double.parseDouble(pasaa1);
            //System.out.println(m1[i]);

}

m2=new double[numI];
for(int i=0;i<numI;i++){
            a=hoja3.getCell(2,2+i);
            pasaa1 =a.getContents().replace(',', '.');
            m2[i]=Double.parseDouble(pasaa1);
            //System.out.println(m2[i]);

}

m3=new double[numI];
for(int i=0;i<numI;i++){
            a=hoja3.getCell(3,2+i);
            pasaa1 =a.getContents().replace(',', '.');
            m3[i]=Double.parseDouble(pasaa1);
            //System.out.println(m3[i]);

}

pAlfa=new double[numI];
for(int i=0;i<numI;i++){
            a=hoja3.getCell(4,2+i);
            pasaa1 =a.getContents().replace(',', '.');
            pAlfa[i]=Double.parseDouble(pasaa1);

            }

pBeta=new double[numI];
for(int i=0;i<numI;i++){
            a=hoja3.getCell(5,2+i);
            pasaa1 =a.getContents().replace(',', '.');
            pBeta[i]=Double.parseDouble(pasaa1);

            }

qAlfa=new double[numI];
for(int i=0;i<numI;i++){
            a=hoja3.getCell(6,2+i);
            pasaa1 =a.getContents().replace(',', '.');
            qAlfa[i]=Double.parseDouble(pasaa1);

            }

qBeta=new double[numI];
for(int i=0;i<numI;i++){
            a=hoja3.getCell(7,2+i);
            pasaa1 =a.getContents().replace(',', '.');
            qBeta[i]=Double.parseDouble(pasaa1);

            }

pVi=new double[numI];
```

```
   for(int i=0;i<numI;i++){
            a=hoja4.getCell(1,2+i);
            pasaa1 =a.getContents().replace(',', '.');
            pVi[i]=Double.parseDouble(pasaa1);

            }

   vRit =new double [numI][numT];
   for(int i=0;i<numI;i++){
            for(int t=0;t<numT;t++  ){
                     a=hoja4.getCell(2+t,2+i);
                     pasaa1 =a.getContents().replace(',', '.');
                     vRit[i][t]=Double.parseDouble(pasaa1);
            }//

pCit =new double [numI][numT];
   for(int i=0;i<numI;i++){
            for(int t=0;t<numT;t++  ){
                     a=hoja5.getCell(1+t,2+i);
                     pasaa1 =a.getContents().replace(',', '.');
                     pCit[i][t]=Double.parseDouble(pasaa1);

                     }
            }//rPrecios Compra

   dEMito=new double [numI][numT][numO];
   GenOcur gn =new GenOcur();
   for(int i=0;i<numI;i++){

                     //double m=4000;  //Es de pruebas


                     double m=0;
            double p=0;
            double q=0;

            for(int o=0;o<numO;o++){
            m = gn.GenerarTriangular(m1[i], m2[i], m3[i]);
            p = gn.generarbetainv(pAlfa[i], pBeta[i]);
            q = gn.generarbetainv(qAlfa[i], qBeta[i]);

if(p == 0 ){

            for(int t=(inidem-1);t<numT;t++){

            dEMito[i][t][o]= 0;
                                                                  }

            }

            else{


            double p_q = (-1)*(p+q);

            double qp = q/p;


            for(int t=(inidem-1);t<numT;t++){
```

```
        String aa = String.valueOf((int)((m*(Math.pow(p_q,
        2)/p)*(Math.exp(p_q*(t+1-
        inidem)))/Math.pow((1+qp*Math.exp(p_q*(t+1-inidem))),
        2))));


    dEMito[i][t][o]=Double.parseDouble(aa);

                                            }

                                    }

                            }

                    }

                    cOkt = new double[numK][numT];
                    for(int k=0; k<numK; k++){ //For 2
                            for(int t=0;t<numT;t++  ){
                                    a=hoja6.getCell(1+t,2+k);
                                    pasaa1
=a.getContents().replace(',', '.');

    cOkt[k][t]=Double.parseDouble(pasaa1);
                            }
                    } // Fin del For 2

                    cCAPrt = new double[numR][numT];
                    for(int r=0; r<numR; r++){ //For 2
                            for(int t=0;t<numT;t++  ){
                                    a=hoja7.getCell(1+t,2+r);
                                    pasaa1
=a.getContents().replace(',', '.');
                                    cCAPrt
[r][t]=Double.parseDouble(pasaa1);
                            }
                    } // Fin del For 2

                    mik =new double[numI][numK];
                    for(int i=0;i<numI;i++){
                            for(int k=0;k<numK;k++  ){
                                    a=hoja8.getCell(1+k,2+i);
                                    pasaa1
=a.getContents().replace(',', '.');
                                    mik
[i][k]=Double.parseDouble(pasaa1);
                            }
                    }

                    nik =new double[numI][numK];
                            for(int i=0;i<numI;i++){
                                    for(int k=0;k<numK;k++  ){

    a=hoja9.getCell(1+k,2+i);
                                            pasaa1
=a.getContents().replace(',', '.');
                                            nik
[i][k]=Double.parseDouble(pasaa1);
                                    }
                            }
```

```
                        rEkr =new double[numK][numR];
                        lt=new int[numK];

                        for(int k=0;k<numK;k++){
                                a=hoja1.getCell(11, 2+k);
                                pasaa1
=a.getContents().replace(',', '.');
                                lt[k]=Integer.parseInt(pasaa1);


                        for(int r=0;r<numR;r++  ){
                                a=hoja10.getCell(1+r,2+k);
                                pasaa1
=a.getContents().replace(',', '.');

   rEkr[k][r]=Double.parseDouble(pasaa1);
                                }
                        }

                        aCit =new double [numI][numT];

                        for(int i=0;i<numI;i++){
                                for(int t=0;t<numT;t++  ){

   a=hoja11.getCell(1+t,2+i);
                                        pasaa1
=a.getContents().replace(',', '.');

   aCit[i][t]=Double.parseDouble(pasaa1);
                                }
                        }

                        kAPrt =new double[numR][numT];

                        for(int r=0;r<numR;r++){
                                for(int t=0;t<numT;t++){

   a=hoja12.getCell(1+t,2+r);
                                        pasaa1
=a.getContents().replace(',', '.');
                                        kAPrt
[r][t]=Double.parseDouble(pasaa1);
                                        }
                        }

                        iNI=new double[numI];
                        cHit=new double[numI][numT];
                        cBit=new double[numI][numT];

                        for(int i=0;i<numI;i++){
                                a=hoja13.getCell(1,1+i);
                                pasaa1
=a.getContents().replace(',', '.');
                        iNI[i]=Double.parseDouble(pasaa1); //
Asigna y convierte a double el valor leido
                        }

                        for(int i=0;i<numI;i++){
                                for(int t=0;t<numT;t++){

   a=hoja13.getCell(2+t,1+i);
```

```
                                        pasaa1
=a.getContents().replace(',', '.');

   cHit[i][t]=Double.parseDouble(pasaa1); // Asigna y convierte
a double el valor leido
                                }
                        }

                        for(int i=0;i<numI;i++){
                        for(int t=0;t<numT;t++){
                                a=hoja16.getCell(1+t,2+i);
                                pasaa1
=a.getContents().replace(',', '.');

   cBit[i][t]=Double.parseDouble(pasaa1);
                                }
                        }

                        pEN1 =new double[numR][numT];

                        for(int r=0;r<numR;r++){
                                for(int t=0;t<numT;t++){

   a=hoja14.getCell(1+t,2+r);
                                        pasaa1
=a.getContents().replace(',', '.');

   pEN1[r][t]=Double.parseDouble(pasaa1);
                                }
                        }

                        pEN2 =new double[numR][numT];

                        for(int r=0;r<numR;r++){
                                for(int t=0;t<numT;t++){

   a=hoja15.getCell(1+t,2+r);
                                        pasaa1
=a.getContents().replace(',', '.');

   pEN2[r][t]=Double.parseDouble(pasaa1);
                                }
                        }

                } catch (BiffException e) {

                        e.printStackTrace();
                } catch (IOException e) {

                        e.printStackTrace();
                }
                }

                public int getTao() {
                        return tao;
                }

                public void setTao(int tao) {
                        this.tao = tao;
                }
```

```
		public int getLa() {
			return La;
		}

		public static void setLa(int la) {
			La = la;
		}

		public static int getLb() {
			return Lb;
		}

		public static void setLb(int lb) {
			Lb = lb;
		}
}
```

Clase para escribir ficheros de datos de entrada al primer modelo matemático

```
package EstocaDET;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.File;
import EstocaDET.*;
	/**
	 *
	 * En esta clase se escriben los datos en fichero .txt
del modelo matemático para la fase I
	 *
	 */
	public class EscribirTXT {

		public EscribirTXT(LeerXML le){
			try{
			File fich =new
File("entradas/datosM1.txt");
			if(fich.exists()==true)
			fich.delete();

			FileWriter fw = new FileWriter(fich,
true);
			PrintWriter w = new PrintWriter(fw);

			String linea ="";
			w.println("data ;");

		//Escritura de La
			linea = "param La:= ";
			linea= linea + (le.getLa());
			linea =linea + " ;";
			w.println(linea); // imprime en el archivo
de datos

		//Escritura de Lb
			linea = "param Lb:= ";
			linea= linea + le.numT;
			linea =linea + " ;";
		w.println(linea); // imprime en el archivo de datos
```

```
   //Escritura de E
            linea = "param tao:= ";
            linea= linea + (le.getTao()+1);
            linea =linea + " ;";
            w.println(linea); // imprime en el archivo de datos

            //Escritura de E
            linea = "param E:= ";
            linea= linea + le.numE;
            linea =linea + " ;";
            w.println(linea); // imprime en el archivo de datos

   // Escribir el conjunto I

            linea = "set I:= ";

for(int i=0;i<le.numI;i++){
                     linea =linea + le.getSKU[i] +" ";

            }
             linea =linea + " ;";
            w.println(linea); // imprime en el archivo de datos

            // Escribir el conjunto K
            linea = "set K:= ";
            for(int k=0;k<le.numK;k++){

                     linea =linea + le.getStroke[k] +" ";

            }
             linea =linea + " ;";
            w.println(linea); // imprime en el archivo de datos

            // Escribir el conjunto R
            linea = "set R:= ";
            for(int r=0;r<le.numR;r++){

                     linea =linea + le.getRecursos[r] +" ";

            }
            linea =linea + " ;";
            w.println(linea); // imprime en el archivo de datos

//Escribir la demanda por escenarios para el primer modelo

            linea = "param D:= ";
            for(int e=0;e<le.numE;e++){

            //Esta esctructura siempre se ha de escribir

linea =linea + "[*,*, " + le.getEscenarios[e] +"]: ";

for(int t=0; t<le.numT;t++){
            linea =linea + le.getPeriodos[t] +" ";
            }
            linea =linea + " :=";

            w.println(linea);//imprime la linea

            linea="";
```

```
//Imprimo los datos de cada cluster

            for(int i=0; i<le.numI;i++){

            linea =linea + le.getSKU[i] +" ";


            for(int t=0; t<le.numT;t++){
            linea = linea + le.dEMite[i][t][e] +" ";
                                        }//Fin For T


w.println(linea);
            linea="";
            }//Fin FOR I

linea =linea + ";";
            w.println(linea);
            }// Fin FOR E

//***********************************************

// FIn demanda por escenarios


            //Escritura del precio de venta del SKU i
            linea = "param PV:= ";
            for(int i=0;i<le.numI;i++){
                        linea = linea + le.getSKU[i] + " " +
le.pVi[i];
                        w.println(linea);
                        linea="";
            }
            linea =linea + ";";
            w.println(linea);

            //Escritura del Inventario Inicial del SKU i
            linea = "param ini:= ";
            for(int i=0;i<le.numI;i++){
                        linea = linea + le.getSKU[i] + " " +
le.iNI[i];
                        w.println(linea);
                        linea="";
            }
            linea =linea + ";";
            w.println(linea);

            //Escritura de laprobabilidad del escenario e
            linea = "param P:= ";
            for(int e=0;e<le.numE;e++){
            linea = linea + le.getEscenarios[e] + " " +
le.pe[e];
                        w.println(linea);
                        linea="";
            }
            linea =linea + ";";
            w.println(linea);

            //Escritura de los lead time
            linea = "param LT:= ";
```

```
            for(int s=0;s<le.numK;s++){
            linea = linea + le.getStroke[s] + " " + le.lt[s];
                      w.println(linea);
                      linea="";
            }
            linea =linea + ";";
            w.println(linea);

            //Escribir Valor Residual del inventario
            linea= "param VR: " ;

            for(int t=0;t<le.numT;t++){

            linea =linea + le.getPeriodos[t] +" ";

            }
            linea =linea + " :=";
                     w.println(linea);
            linea="";

            for(int i=0; i<le.numI;i++){
                     linea =linea + le.getSKU[i] +" ";

            for(int t=0; t<le.numT;t++){
                     linea = linea + le.vRit[i][t] +" ";
                     }

            w.println(linea);
            linea="";

            }
            w.println(";");

//Finaliza Escritura de valor residual del inventario

   //Escribir Precio de Compra


            linea= "param PC: " ;

   for(int t=0;t<le.numT;t++){

            linea =linea + le.getPeriodos[t] +" ";

            }

            linea =linea + " :=";

            w.println(linea);
            linea="";


   for(int i=0; i<le.numI;i++){

            linea =linea + le.getSKU[i] +" ";


   for(int t=0; t<le.numT;t++){

            linea = linea + le.pCit[i][t] +" ";
```

```
                    }

            w.println(linea);

            linea="";
                                                                    }

            w.println(";");
   //Finaliza Escritura de Precio de compra

   //Escribir Coste de operacion de un stroke
            linea= "param CO: " ;


   for(int t=0;t<le.numT;t++){

            linea =linea + le.getPeriodos[t] +" ";
            }
                                                                    linea
=linea + " :=";

            w.println(linea);
                                                        linea="";


   for(int k=0; k<le.numK;k++){

            linea =linea + le.getStroke[k] +" ";

for(int t=0; t<le.numT;t++){

            linea = linea + le.cOkt[k][t] +" ";

            }

   w.println(linea);
                                            linea="";


}
            w.println(";");

//Finaliza Escritura de Coste de operacion de un stroke

//Escribir Coste reserva de capacidad


                    linea= "param COSTKAP: " ;


            for(int t=0;t<le.numT;t++){


                    linea =linea + le.getPeriodos[t] +" ";
                                                }

                    linea =linea + " :=";

   w.println(linea);
```

```
                linea="";

        for(int r=0; r<le.numR;r++){
                linea =linea + le.getRecursos[r] +" ";

                for(int t=0; t<le.numT;t++){
                linea = linea + le.cCAPrt[r][t] +" ";
        }
        w.println(linea);
        linea="";
        }
        w.println(";");
//Finaliza Escritura de coste de reserva de capacidad

//Escribir MATRIZ MIK

        linea= "param M: " ;

        for(int k=0;k<le.numK;k++){
                linea =linea + le.getStroke[k] +" ";
        }
        linea =linea + " :=";
        w.println(linea);
        linea="";

        for(int i=0; i<le.numI;i++){
        linea =linea + le.getSKU[i] +" ";

        for(int k=0; k<le.numK;k++){
                linea = linea + le.mik[i][k] +" ";
        }
        w.println(linea);
        linea="";
        }
```

```
            w.println(";");

//Finaliza Escritura de M

                                                    //Escribir
MATRIZ N

            linea= "param N: " ;

            for(int k=0;k<le.numK;k++){

            linea =linea + le.getStroke[k] +" ";

            }
            linea =linea + " :=";
            w.println(linea);
            linea="";

            for(int i=0; i<le.numI;i++){

                     linea =linea + le.getSKU[i] +" ";


                     for(int k=0; k<le.numK;k++){

                              linea = linea + le.nik[i][k] +"
";                                                          }


                              w.println(linea);
                              linea="";
   }
                     w.println(";");
            //Finaliza Escritura de N

   //Escribir coeficientes tecnologicos


                     linea= "param RE: " ;
```

```
                    for(int r=0;r<le.numR;r++){

                    linea =linea + le.getRecursos[r] +" ";
                }
                linea =linea + " :=";
                w.println(linea);
                linea="";
                for(int k=0; k<le.numK;k++){
                    linea =linea + le.getStroke[k] +" ";

                    for(int r=0;r<le.numR;r++){
                        linea = linea + le.rEkr[k][r] +"
";                                          }
                    w.println(linea);
                        linea="";
                }
                    w.println(";");
    //Finaliza Escritura de coeficientes tecnologicos
    //Escribir AC
                    linea= "param AC: " ;

                for(int t=0;t<le.numT;t++){

                    linea =linea + le.getPeriodos[t] +" ";
                }
                linea =linea + " :=";
                w.println(linea);
                linea="";
```

```
            for(int i=0; i<le.numI;i++){
                    linea =linea + le.getSKU[i] +" ";

                    for(int t=0; t<le.numT;t++){
                    linea = linea + le.aCit[i][t] +" ";
                    }
                    w.println(linea);
                    linea="";
                                                        }
            w.println(";");
//Finaliza Escritura de AC
//Escribir BAckOrder
            linea= "param CB: " ;
                                for(int t=0;t<le.numT;t++){
            linea =linea + le.getPeriodos[t] +" ";
                    }
                    linea =linea + " :=";
w.println(linea);
                    linea="";

                    for(int i=0; i<le.numI;i++){
                    linea =linea + le.getSKU[i] +" ";

            for(int t=0; t<le.numT;t++){
                    linea = linea + le.cBit[i][t] +" ";
```

```
                    }
w.println(linea);
                    linea="";
}
                    w.println(";");
          //Finaliza Escritura de ABackOrders
          //Escribir Coste de almacenamiento
          linea= "param CH: " ;
                              for(int t=0;t<le.numT;t++){
linea =linea + le.getPeriodos[t] +" ";
                              }
          linea =linea + " :=";
          w.println(linea);
          linea="";
          for(int i=0; i<le.numI;i++){
                    linea =linea + le.getSKU[i] +" ";
          for(int t=0; t<le.numT;t++){
                    linea = linea + le.cHit[i][t] +" ";
                    }
                    w.println(linea);
                    linea="";
                    }
          w.println(";");
          //Finaliza Escritura de Coste almacenamiento
```

```
            //Escribir Capacidad máxima ofertada por el
proveedor
                        linea= "param KAP: " ;

                        for(int t=0;t<le.numT;t++){

                        linea =linea + le.getPeriodos[t] +" ";}

                        linea =linea + " :=";

                        w.println(linea);
                        linea="";

            for(int r=0; r<le.numR;r++){
                        linea =linea + le.getRecursos[r] +" ";


                        for(int t=0; t<le.numT;t++){

                                    linea = linea + le.kAPrt[r][t] +"
";}
                        w.println(linea);
                        linea="";
                        }
                        w.println(";");
//Finaliza Escritura de Capacidad máxima ofertada por el
proveedor
            //Escribir Coste Penalidad positiva

            linea= "param PEN2: " ;

            for(int t=0;t<le.numT;t++){

                        linea =linea + le.getPeriodos[t] +" ";}

                        linea =linea + " :=";
                        w.println(linea);
```

```
                    linea="";

                for(int r=0; r<le.numR;r++){
                    linea =linea + le.getRecursos[r] +" ";

                    for(int t=0; t<le.numT;t++){

                        linea = linea + le.pEN2[r][t] +"
";}

                    w.println(linea);
                    linea="";
                    }
                    w.println(";");

                linea= "param PEN1: " ;

                for(int t=0;t<le.numT;t++){

                    linea =linea + le.getPeriodos[t] +" ";}

                        linea =linea + " :=";
                        w.println(linea);
                        linea="";

                    for(int r=0; r<le.numR;r++){
                    linea =linea + le.getRecursos[r] +" ";

                    for(int t=0; t<le.numT;t++){

                        linea = linea + le.pEN1[r][t] +"
"; }
                    w.println(linea);
                    linea="";
                    }
```

```
                        w.println(";");


                        linea ="";
                        w.println("end;");

                w.close();}

// Para no imprimir más en el archivo
   // Fin Try


                catch(Exception eP ){

                eP.printStackTrace();

   }// Fin Catch

   }//Fin del método

}//Fin de la clase
```

Clase de escritura de data para modelo Fase II

```
package EstocaDET;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.File;
import EstocaDET.*;
/**
 *
 * En esta clase se escriben los datos que incurren a la fase
II. Se le pasarán los parámetros
 * que vienen de la fase I a la fase II como lo es el
inventario, la capacidad contratada, los
 * productos generados por cada stroke que resultaran en las
Recepciones Planificadas.
 *
 */
public class EscribirTXT2 {


public EscribirTXT2(LeerXML le, SolveM1 model, double[] inv,
double[] bak, String nombre, int o){

   try{

                File fich =new File(nombre);
                if(fich.exists()==true)
                fich.delete();

                FileWriter fw = new FileWriter(fich, true);
                PrintWriter w = new PrintWriter(fw);

                String linea ="";
                w.println("data ;");

//Desde aquí se comienza a escribir el fichero de datos
```

```
//Escritura de La
         linea = "param La:= ";
         linea= linea + (le.getTao()+1);
         linea =linea + " ;";
         w.println(linea); // imprime en el archivo de datos

//Escritura de Lb
         linea = "param Lb:= ";
         linea= linea + le.numT;
         linea =linea + " ;";
         w.println(linea); // imprime en el archivo de datos


// Escribir el conjunto I

         linea = "set I:= ";
          for(int i=0;i<le.numI;i++){

                   linea =linea + le.getSKU[i] +" ";}
          linea =linea + " ;";
         w.println(linea); // imprime en el archivo de datos


         // Escribir el conjunto K
         linea = "set K:= ";
          for(int k=0;k<le.numK;k++){

                  linea =linea + le.getStroke[k] +" ";}
          linea =linea + " ;";
         w.println(linea); // imprime en el archivo de datos


         // Escribir el conjunto R
         linea = "set R:= ";
         for(int r=0;r<le.numR;r++){

                  linea =linea + le.getRecursos[r] +" ";}

         linea =linea + " ;";
         w.println(linea); // imprime en el archivo de datos


         //Escribir Demanda REal
         linea= "param D: " ;
         for(int t=le.tao;t<le.numT;t++){

                  linea =linea + le.getPeriodos[t] +" ";}
                                    linea =linea + " :=";

         w.println(linea);

         linea="";


         for(int i=0; i<le.numI;i++){

                  linea =linea + le.getSKU[i] +" ";
```

```
   for(int t=le.tao; t<le.numT;t++){

   linea = linea + le.dEMito[i][t][o] +" ";}

//System.out.println(le.dEMito[i][t][o]);
            w.println(linea);
            linea="";
            }

            w.println(";");


//Finaliza Escritura de la Demanda Real

            //Escritura del precio de venta del SKU i
                     linea = "param PV:= ";
                     for(int i=0;i<le.numI;i++){
                              linea = linea + le.getSKU[i] + "
" + le.pVi[i];
                              w.println(linea);
                              linea="";}
            linea =linea + ";";
            w.println(linea);

            //Escritura del Inventario Inicial del SKU i
            linea = "param ini:= ";
            for(int i=0;i<le.numI;i++){
                     linea = linea + le.getSKU[i] + " " +
inv[i];
                      w.println(linea);
                      linea="";}
            linea =linea + ";";
            w.println(linea);

//Escritura del BackOrder Inicial del SKU i, si las hay
            linea = "param BAKIN:= ";
            for(int i=0;i<le.numI;i++){
                      linea = linea + le.getSKU[i] + " " +
bak[i];
                      w.println(linea);
                      linea="";}
            linea =linea + ";";
            w.println(linea);

            //Escritura de los lead time
            linea = "param LT:= ";
            for(int s=0;s<le.numK;s++){
linea = linea + le.getStroke[s] + " " + le.lt[s];
                      w.println(linea);
                      linea="";}
            linea =linea + ";";
            w.println(linea);

            //Escribir Valor Residual del inventario
            linea= "param VR: " ;

             for(int t=le.tao;t<le.numT;t++){

                              linea =linea + le.getPeriodos[t]
+" ";
                    }
```

```
                    linea =linea + " :=";

             w.println(linea);
            linea="";

            for(int i=0; i<le.numI;i++){
                              linea =linea + le.getSKU[i] +" ";

            for(int t=le.tao; t<le.numT;t++){
            linea = linea + le.vRit[i][t] +" ";
                              }

                     w.println(linea);
                     linea="";

                     }
            w.println(";");
//Finaliza Escritura de valor residual del inventario

//*****
//Escribir Precio de Compra


            linea= "param PC: " ;

            for(int t=le.tao;t<le.numT;t++){

            linea =linea + le.getPeriodos[t] +" ";


            }

            linea =linea + " :=";


            w.println(linea);
            linea="";


            for(int i=0; i<le.numI;i++){

            linea =linea + le.getSKU[i] +" ";


            for(int t=le.tao; t<le.numT;t++){

            linea = linea + le.pCit[i][t] +" ";
                              }


            w.println(linea);

            linea="";

            }
            w.println(";");
//Finaliza Escritura de Precio de compra


//Escribir Coste de operacion de un stroke
```

```
            linea= "param CO: " ;

            for(int t=le.tao;t<le.numT;t++){
            linea =linea + le.getPeriodos[t] +" ";
            }
            linea =linea + " :=";

            w.println(linea);
            linea="";

            for(int k=0; k<le.numK;k++){
            linea =linea + le.getStroke[k] +" ";

            for(int t=le.tao; t<le.numT;t++){
            linea = linea + le.cOkt[k][t] +" ";
                     }

            w.println(linea);
            linea="";

                     }
            w.println(";");
//Finaliza Escritura de Coste de operacion de un stroke

//Escribir MATRIZ MIK

linea= "param M: " ;

for(int k=0;k<le.numK;k++){
            linea =linea + le.getStroke[k] +" ";

            }
            linea =linea + " :=";

            w.println(linea);
```

```
linea="";

for(int i=0; i<le.numI;i++){
          linea =linea + le.getSKU[i] +" ";


for(int k=0; k<le.numK;k++){
          linea = linea + le.mik[i][k] +" ";
          }
          w.println(linea);
          linea="";
}
w.println(";");
//Finaliza Escritura de M

//Escribir MATRIZ N

linea= "param N: " ;

for(int k=0;k<le.numK;k++){

linea =linea + le.getStroke[k] +" ";

}
linea =linea + " :=";

w.println(linea);
linea="";

for(int i=0; i<le.numI;i++){

linea =linea + le.getSKU[i] +" ";

for(int k=0; k<le.numK;k++){

linea = linea + le.nik[i][k] +" ";
```

```
}

w.println(linea);
linea="";

}
w.println(";");
//Finaliza Escritura de N

//Escribir coeficientes tecnologicos

linea= "param RE: " ;

for(int r=0;r<le.numR;r++){

linea =linea + le.getRecursos[r] +" ";

}
linea =linea + " :=";

w.println(linea);
linea="";

for(int k=0; k<le.numK;k++){

linea =linea + le.getStroke[k] +" ";


for(int r=0;r<le.numR;r++){

linea = linea + le.rEkr[k][r] +" ";

}

w.println(linea);
linea="";

}
```

```
            w.println(";");
//Finaliza Escritura de coeficientes tecnologicos


//Escribir AC

linea= "param AC: " ;


for(int t=le.tao;t<le.numT;t++){


linea =linea + le.getPeriodos[t] +" ";


}
linea =linea + " :=";


w.println(linea);
linea="";


for(int i=0; i<le.numI;i++){

linea =linea + le.getSKU[i] +" ";


for(int t=le.tao; t<le.numT;t++){

linea = linea + le.aCit[i][t] +" ";

}


w.println(linea);

linea="";


}
```

```
w.println(";");
//Finaliza Escritura de AC

//Escribir BAckOrder

linea= "param CB: " ;


for(int t=le.tao;t<le.numT;t++){

linea =linea + le.getPeriodos[t] +" ";

}
linea =linea + " :=";


w.println(linea);
linea="";


for(int i=0; i<le.numI;i++){

linea =linea + le.getSKU[i] +" ";


for(int t=le.tao; t<le.numT;t++){

linea = linea + le.cBit[i][t] +" ";

}


w.println(linea);

linea="";
}
w.println(";");
//Finaliza Escritura de ABackOrders
```

```
//Escribir Coste de almacenamiento


linea= "param CH: " ;


for(int t=le.tao;t<le.numT;t++){

linea =linea + le.getPeriodos[t] +" ";
                                        }

linea =linea + " :=";
w.println(linea);
linea="";


for(int i=0; i<le.numI;i++){

linea =linea + le.getSKU[i] +" ";

for(int t=le.tao; t<le.numT;t++){

linea = linea + le.cHit[i][t] +" ";

}

w.println(linea);
linea="";

}
w.println(";");
//Finaliza Escritura de Coste almacenamiento
//Escribir Capacidad contratada previamente

linea= "param CAP: " ;

for(int t=le.tao;t<le.numT;t++){
```

```
                linea =linea + le.getPeriodos[t] +" ";


                }

                linea =linea + " :=";

                w.println(linea);

                linea="";


                for(int r=0; r<le.numR;r++){

                linea =linea + le.getRecursos[r] +" ";


                for(int t=le.tao; t<le.numT;t++){

                if (le.cCAPrt[r][t]==0){
                linea = linea + le.kAPrt[r][t]  +" ";
                }else{
                linea = linea + model.kap1[r][t]  +" ";
                                    }

                          }


                w.println(linea);

                linea="";


                }

                w.println(";");

//Finaliza Escritura de Capacidad máxima ofertada por el
proveedor

//Escribir Coste Penalidad positiva

                linea= "param PEN2: " ;


                for(int t=le.tao;t<le.numT;t++){


                linea =linea + le.getPeriodos[t] +" ";


                }

                linea =linea + " :=";


                w.println(linea);

                linea="";
```

```
for(int r=0; r<le.numR;r++){
        linea =linea + le.getRecursos[r] +" ";

for(int t=le.tao; t<le.numT;t++){
        linea = linea + le.pEN2[r][t] +" ";
        }
        w.println(linea);
        linea="";
        }
        w.println(";");
//Finaliza coste de penalidad positiva
//Escribir Coste Penalidad negativa
        linea= "param PEN1: " ;
for(int t=le.tao;t<le.numT;t++){
        linea =linea + le.getPeriodos[t] +" ";
        }
        linea =linea + " :=";
        w.println(linea);
        linea="";
for(int r=0; r<le.numR;r++){
        linea =linea + le.getRecursos[r] +" ";
for(int t=le.tao; t<le.numT;t++){
        linea = linea + le.pEN1[r][t] +" ";
```

```
                    }

            w.println(linea);
            linea="";

            }
            w.println(";");
//Finaliza coste de penalidad negativa
//Escribir Recepciones planificadas

            linea= "param RPL: " ;

for(int t=le.tao;t<le.numT;t++){

            linea =linea + le.getPeriodos[t] +" ";

            }
            linea =linea + " :=";

            w.println(linea);
            linea="";

for(int i=0; i<le.numI;i++){

            linea =linea + le.getSKU[i] +" ";
for(int t=le.tao; t<le.numT;t++){

            linea = linea + model.RPL[i][t] +" ";

            }

            w.println(linea);

            linea="";
```

```
            }

            w.println(";");

   //Finaliza Escritura de ABackOrders


            linea ="";
            w.println("end;");

            w.close(); // Para no imprimir mas en el archivo

            } // Fin Try

            catch(Exception eP ){
            eP.printStackTrace();

            }// Fin Catch

}//Fin del método
}//Fin de la clase
```

Lectura de XML

```
package EstocaDET;

import java.io.File;
import java.util.ArrayList;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;
import com.sun.org.apache.xalan.internal.xsltc.compiler.Parser;


import LectorXML.ActivaCompra;
import LectorXML.BOM;
import LectorXML.CoefTecno;
import LectorXML.CosCapacidad;
import LectorXML.CosteBackOrder;
import LectorXML.CosteInventory;
import LectorXML.CosteStroke;
import LectorXML.Data;
import LectorXML.DemBass;
import LectorXML.DemandaEs;
import LectorXML.DemandaRe;
import LectorXML.InventarioIncial;
import LectorXML.Leadtime;
import LectorXML.PenalidadMAS;
import LectorXML.PenalidadMENOS;
import LectorXML.PrecioCompra;
import LectorXML.PrecioProd;
import LectorXML.ProbabilidadEsc;
import LectorXML.RBOM;
import LectorXML.ResidualInv;
import LectorXML.UpperBounCapacidad;

/**
* En esta clase se leen los datos tomados de XML que guardados
en variables para su posteriortratamiento
 *
 */
public class LeerXML {
```

```
   // Declaración de atributos del objeto: problema a resolver

   //Número de elementos de cada conjunto
   public static int numI;
   public static int numT;
   public static int numK;
   public static int numR;
   public static int numE;
   public static int tao;
   public static int numO;

   //Parámetros del modelo de difusión
   public static double[] pi;
   public static double[] qi;
   public static double[] mEi;
   public static double[] mRi;
   public static double[] errori;

   //Elementos que forman los conjuntos
   public static String []getSKU ;
   public static int[]getEscenarios ;
   public static int[]getOcurrencias ;
   public static String []getStroke ;
   public static String []getRecursos;
   public static int[]getPeriodos;

   /**
* Este parámetro es la demanda del SKU i en el tiempo t
generada para el escenario e
    */
public static double [][][] dEMite;//Demanda del SKU i en t en
el escenario e

   /**
    * Este parámetro es la demanda real del  SKU i en el tiempo
t
    *
    */
public static double [][][] dEMito;//Demandas generadas por
ocurrencias
   public static double [][] dEMit;
// TODO Colocar la otra dimencion a la Variable dEMit la de las
ocurrencias

   /**
    * Este parámetro es el Precio de venta del producto i
    */
   public static double [] pVi;

   /**
* Este parametro es el Valor Residual del inventario del SKU i
en el periodo t
    */
   public static double [][] vRit;

   /**
* Este parámetro es el precio al cual se compra la materia
prima y los componentes
    */
```

```
   public static double [][] pCit;

   /**
* Este parámetro se refiere al coste de un STROKE k en el
periodo t
    */
   public static double [][] cOkt;

   /**
* Parámetro del coste de reserva de capacidad del recurso r en
el periodo t
    */
   public static double [][] cCAPrt;

   /**
    * Matriz de componentes i requeridos para un stroke k
    */
   public static double [][] mik;

   /**
    * Matriz de componentes i generados por un stroke k
    */
   public static double [][] nik;

   /**
* Coeficientes Tecnológicos del consumo del recurso por parte
del stroke k
    */
   public static double [][] rEkr;

   /**
    * Activación de SKU a comprar en dimensiones i e t
    */
   public static double [][] aCit;

   /**
* Capacidad máxima ofrecida por un proveedor de capacidad de
recursos
    */
   public static double [][] kAPrt;

   /**
    * Probabilidad de ocurrencia del escenario e
    */
   public static double [] pe;


   public static double [] iNI;//Inventario Inicial
   public static int La;//Periodo Inicial
   public static int Lb;//Periodo Final
   public static int []lt; //Lead Time del stroke K

   public static double [][] pEN1; //Penalidad
   public static double [][] pEN2; //Penalidad

   public static double [][] cHit; // Coste de Almacenamiento
   public static double [][] cBit; // Coste de REtrasos

   /**
    * Este método lee un XML de entrada para inicializar el
problema
```

```
    * @param sEntrada
    * Las entradas son un XML con la informacion de un problema
especifico a resolver
    */
            public static void LeerXML1(String sEntrada){
            try{

   // Se cosntruye el procedimiento para leer el XML
JAXBContext jaxbContext = JAXBContext.newInstance("LectorXML");

//Debe tener el nombre
Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();


Data datos = (Data) unmarshaller.unmarshal(new File(sEntrada));
      // Entrada es el nombre del Fichero XML

      // Número de elementeos de cada conjunto

            numI=datos.getNumI();
            numK=datos.getNumK();
            numR=datos.getNumR();
            numE=datos.getNumE();
            numT=datos.getNumT();
            numO=datos.getNumO();
            tao=datos.getTao();


      // Inicializacion de todos los parámetros

      // Elementos de cada conjunto del problema

      // Se toma el valor de cada uno de los periodos

            //Lectura de los elementos del conjunto t
            //*****************************************
            getPeriodos=new int[numT];
            ArrayList <String> periodos =

   (ArrayList)datos.getConjuntoPeriodos();
            for(int i=0;i<periodos.size();i++){

                     String time = periodos.get(i);

                     getPeriodos[i]=Integer.parseInt(time);
            }//Fin lectura
            //*********************************************



            //Lectura de los elementos del conjunto E
            //*****************************************
            getEscenarios=new int[numE];
ArrayList <String> escenarios =
(ArrayList)datos.getConjuntoEscenarios();
            for(int i=0;i<escenarios.size();i++){

                     String esene = escenarios.get(i);
                     getEscenarios[i]=Integer.parseInt(esene);
            }//Fin lectura
```

```
            //*********************************************



            //-----------------------------------------------
            // Lectura de elementos
            //-----------------------------------------------

            getOcurrencias=new int[numO];
ArrayList <String> ocurriendo =
(ArrayList)datos.getConjuntoEventos();
            for(int i=0;i<ocurriendo.size();i++){

                    String eseni = ocurriendo.get(i);
                    getOcurrencias[i]=Integer.parseInt(eseni);
            }//Fin lectura


            //*********************************************

//Lectura de los elementos del conjunto R
            //*******************************************
            getRecursos=new String [numR];
ArrayList <String> recursos
=(ArrayList)datos.getConjuntoRecursos();
            for(int i=0;i<recursos.size();i++){

                    String resource = recursos.get(i);

                    getRecursos[i]=resource;
            }//Fin lectura
            //*********************************************

            //Lectura de los elementos del conjunto K
            //*******************************************
            getStroke=new String[numK];
ArrayList <String> strokes=
(ArrayList)datos.getCojuntoStrokes();
            for(int i=0;i<strokes.size();i++){

                    String st = strokes.get(i);

                    getStroke[i]=st;
            }//Fin lectura
   //*********************************************

            //Lectura de los elementos del conjunto I
            //*******************************************
            getSKU=new String[numI];
ArrayList<String> skuses = (ArrayList) datos.getConjuntoSKU();
            for(int i=0;i<skuses.size();i++){

                    String sk = skuses.get(i);

                    getSKU[i]=sk;
            }//Fin lectura
            //*********************************************

            /*
* LA LECTURA DE LA DEMANDA PUEDE SER DE MANERA MANUAL A PARTIR
DE UN XML QUE TRAE CONSIGO LOS ESCENARIOS Ó EL CASO En DONDE A
```

```
PARTIR DE UNOS PARÁMETROS SE GENERAN DE MANERA AUTOMATICA LOS
ESCENARIOS
            */

            pe=new double[numE];
            dEMite=new double[numI][numT][numE];//Demanda
estocástica
            dEMito=new double[numI][numT][numO];//Demanda
estocástica
            //dEMit=new double [numI][numT];//Demanda Real

            //Lectura de escenarios generados manualmente

            //Lectura de la demanda por escenario y periodos
            //*********************************************

ArrayList<DemandaEs> demesto = (ArrayList)datos.getDemEsS();

            for(int i=0;i<demesto.size();i++){

                    //Se crea un vector demanda auxiliar
                    DemandaEs demEs = demesto.get(i);

                    //Inicio Lectura primer elemento del
conjunto P
                    int idenSKU=0;
                    for(int ii=0; ii<skuses.size(); ii++){

                            String SKUAux = skuses.get(ii);

            //En esta parte Cambiar la obtencion de getI
                            if(demEs.getI().compareTo(SKUAux)==0){
                            idenSKU = ii;
                            }
            }//Termina primer elemento del indice del conjunto
i
                    int idenPer=0;
                    for(int t=0; t<periodos.size(); t++){

                    String TAux = periodos.get(t);


   if(demEs.getT().compareTo(TAux)==0){
                                    idenPer = t;
                            }
            }//Termina primer elemento del indice del conjunto
t
                    int idenESt=0;
                    for(int e=0; e<escenarios.size(); e++){

                            String EAux = escenarios.get(e);

            //En esta parte Cambiar la obtencion de getI
                            if(demEs.getE().compareTo(EAux)==0){
                            idenESt = e;
                                    }
            }//Termina primer elemento del indice del conjunto
t
```

```
dEMite[idenSKU][idenPer][idenESt]=demEs.getDESite();

            }//Fin lectura de la demanda por escenarios

            //Lectura de la demanda Real
            //*****************************************

            //Lectura de la demanda por escenario y periodos
            //*********************************************
ArrayList<DemandaRe> demesti = (ArrayList)datos.getDemRealS();

for(int i=0;i<demesti.size();i++){
            //Se crea un vector demanda auxiliar
            DemandaRe demas =demesti.get(i);
            //Inicio Lectura primer elemento del conjunto P

            int idenSKU=0;
            for(int ii=0; ii<skuses.size(); ii++){

                     String SKUAux = skuses.get(ii);

            //En esta parte Cambiar la obtencion de getI
                     if(demas.getI().compareTo(SKUAux)==0){
                              idenSKU = ii;
                     }
   }//Termina primer elemento del indice del conjunto i

            int idenPer=0;
            for(int t=0; t<periodos.size(); t++){

                     String TAux = periodos.get(t);

                     if(demas.getT().compareTo(TAux)==0){
                              idenPer = t;
                     }
}
//Termina primer elemento del indice del conjunto t


            int idenEvento=0;
            for(int o=0; o<ocurriendo.size(); o++){

                     String EAux = ocurriendo.get(o);

   //En esta parte Cambiar la obtencion de getI
                     if(demas.getO().compareTo(EAux)==0){
                              idenEvento = o;
                     }
            }
//Termina primer elemento del indice del conjunto t

   dEMito[idenSKU][idenPer][idenEvento]=demas.getDito();

   }//Fin lectura de la demanda por escenarios


ArrayList<ProbabilidadEsc> probil=
(ArrayList)datos.getProbabilidades();
   for(int i=0;i<probil.size();i++){
   //Se crea un vector demanda auxiliar
```

```
            ProbabilidadEsc au = probil.get(i);
            String ss = escenarios.get(i);
            pe[i]=au.getPe();
   }//Fin lectura de precios de venta
   //
Si esto da error hacerlo como en el inventario o el leadtime
   //*********************************************


        // FIN DE LA LECTURA DE LOS VALORES DE DEMANDA


        //Lectura de los valores residuales de inventario
        //*****************************************
            vRit=new double[numI][numT];//Valor residual de inv
ArrayList<ResidualInv> valorResiduales =
(ArrayList)datos.getValorResidualI();
   for(int i=0;i<valorResiduales.size();i++){


//Se crea un vector demanda auxiliar
            ResidualInv reside = valorResiduales.get(i);


//Inicio Lectura primer elemento del conjunto P
            int idenSKU=0;
            for(int ii=0; ii<skuses.size(); ii++){

                    String SKUAux = skuses.get(ii);

   //En esta parte Cambiar la obtencion de getI
                    if(reside.getI().compareTo(SKUAux)==0){
                            idenSKU = ii;
                    }
            }//Termina primer elemento del indice del conjunto
i
            int idenPer=0;
            for(int t=0; t<periodos.size(); t++){

                    String TAux = periodos.get(t);

            //En esta parte Cambiar la obtencion de getI
                    if(reside.getT().compareTo(TAux)==0){
                            idenPer = t;
                    }
            }//Termina primer elemento del indice del conjunto
t
            vRit[idenSKU][idenPer]=reside.getVRit();

   }//Fin lectura de los valores residuales


   //Lectura de los precios de compra
   //*****************************************
   pCit=new double[numI][numT];//Precio de compra de SKU
ArrayList<PrecioCompra> precios =
(ArrayList)datos.getPrecioCompras();
   for(int i=0;i<precios.size();i++){
   //Se crea un vector demanda auxiliar
            PrecioCompra preciocompras = precios.get(i);
```

```
   //Inicio Lectura primer elemento del conjunto P
            int idenSKU=0;
            for(int ii=0; ii<skuses.size(); ii++){

                     String SKUAux = skuses.get(ii);

   //En esta parte Cambiar la obtencion de getI
            if(preciocompras.getI().compareTo(SKUAux)==0){
                     idenSKU = ii;
                     }
            }//Termina primer elemento del indice del conjunto
i
            int idenPer=0;
            for(int t=0; t<periodos.size(); t++){

            String TAux = periodos.get(t);

                     //En esta parte Cambiar la obtencion de
getI

   if(preciocompras.getT().compareTo(TAux)==0){
                     idenPer = t;
                     }
            }//Termina primer elemento del indice del conjunto
t

   pCit[idenSKU][idenPer]=preciocompras.getPCit();

            }//Fin lectura de los precios de compra
            //*****************************************


            //Lectura de los activacion de compras
            //*****************************************
            aCit=new double[numI][numT];//Activar compra
ArrayList<ActivaCompra> activarcompra =
(ArrayList)datos.getCompras();
            for(int i=0;i<activarcompra.size();i++){

                     //Se crea un vector demanda auxiliar

                     ActivaCompra aComp = activarcompra.get(i);

                     //Inicio Lectura primer elemento del
conjunto P
                     int idenSKU=0;
                     for(int ii=0; ii<skuses.size(); ii++){

                              String SKUAux = skuses.get(ii);

                     //En esta parte cambiar la obtencion de
getI

   if(aComp.getI().compareTo(SKUAux)==0){
                                       idenSKU = ii;
                              }
                     }//Termina primer elemento del indice del
conjunto i
                     int idenPer=0;
                     for(int t=0; t<periodos.size(); t++){
```

```
                                String TAux = periodos.get(t);

                        //En esta parte cambiar la obtencion de
getI

   if(aComp.getT().compareTo(TAux)==0){
                                        idenPer = t;
                                }
                        }//Termina primer elemento del indice del
conjunto t
                        aCit[idenSKU][idenPer]=  aComp.getACit();
                }//Fin lectura de activacion de compras
                //*****************************************


                //Lectura Coste de Strokes
                //*****************************************
                cOkt=new double[numK][numT];//Coste operacion
stroke
ArrayList<CosteStroke> cStroke =
(ArrayList)datos.getCosStrokes();
                for(int i=0;i<cStroke.size();i++){

                        //Se crea un vector demanda auxiliar

                        CosteStroke cosS = cStroke.get(i);

                        //Inicio Lectura primer elemento del
conjunto P
                        int idenStroke=0;
                        for(int k=0; k<strokes.size(); k++){

                                String StrokeAux =
strokes.get(k);
                        //En esta parte cambiar la obtencion de
getI

   if(cosS.getK().compareTo(StrokeAux)==0){
                                        idenStroke = k;
                                }
                        }//Termina primer elemento del indice del
conjunto i
                        int idenPer=0;
                        for(int t=0; t<periodos.size(); t++){

                                String TAux = periodos.get(t);

                        //En esta parte cambiar la obtencion de
getI

   if(cosS.getT().compareTo(TAux)==0){
                                        idenPer = t;
                                }
                        }//Termina primer elemento del indice del
conjunto t
                        cOkt[idenStroke][idenPer]= cosS.getCOkt();

                }//Fin lectura de coste de strokes
                //*****************************************
```

```
            //Lectura Coste de capacidad
            //*****************************************
            cCAPrt=new double[numR][numT];//Coste compra
capacidad
ArrayList<CosCapacidad> ccaP =
(ArrayList)datos.getCostesCapacidad();
            for(int i=0;i<ccaP.size();i++){

            //Se crea un vector demanda auxiliar

                     CosCapacidad cosCap = ccaP.get(i);

            //Inicio Lectura primer elemento del conjunto P
                     int idenRecursis=0;

                     for(int r=0; r<recursos.size(); r++){

                              String recursiAux =
recursos.get(r);
            //En esta parte cambiar la obtencion de getI

   if(cosCap.getR().compareTo(recursiAux)==0){
                                       idenRecursis = r;
                              }
                     }
            //Termina primer elemento del indice del conjunto i

                     int idenPer=0;
                     for(int t=0; t<periodos.size(); t++){

                              String TAux = periodos.get(t);

            //En esta parte Cambiar la obtencion de getI

   if(cosCap.getT().compareTo(TAux)==0){
                                       idenPer = t;
                              }
                     }
            //Termina primer elemento del indice del conjunto t

                     cCAPrt[idenRecursis][idenPer]=
cosCap.getCCAPrt();

            }//Fin lectura de coste de strokes
            //*****************************************


            //Lectura Upper Bound Capacidad
            //*****************************************
            kAPrt=new double[numR][numT];////Upper Boun
capacidad
ArrayList<UpperBounCapacidad> kkap =
(ArrayList)datos.getLimiteRecursos();
            for(int i=0;i<kkap.size();i++){

                     //Se crea un vector demanda auxiliar

                     UpperBounCapacidad uper = kkap.get(i);

                     //Inicio Lectura primer elemento del
conjunto P
```

```
                        int idenRecursis=0;

                        for(int r=0; r<recursos.size(); r++){

                                String recursiAux =
recursos.get(r);
                        /En esta parte Cambiar la obtencion de
getI

   if(uper.getR().compareTo(recursiAux)==0){
                                        idenRecursis = r;
                                }
                        }//Termina primer elemento del indice del
conjunto i
                        int idenPer=0;
                        for(int t=0; t<periodos.size(); t++){

                                String TAux = periodos.get(t);

                                //En esta parte Cambiar la
obtencion de getI

   if(uper.getT().compareTo(TAux)==0){
                                        idenPer = t;
                                }
                        }//Termina primer elemento del indice del
conjunto t
                        kAPrt[idenRecursis][idenPer]=
uper.getKAPrt();
                }//Fin lectura de UpperBound
                //*****************************************



                //Lectura Penalidad 1
                //*****************************************
                pEN1=new double[numR][numT];//Penalidad Extra
ArrayList<PenalidadMAS> penalidadmas =
(ArrayList)datos.getPenalidadesMas();
                for(int i=0;i<penalidadmas.size();i++){

                        //Se crea un vector demanda auxiliar

                        PenalidadMAS pnali = penalidadmas.get(i);

                        //Inicio Lectura primer elemento del
conjunto P
                        int idenRecursis=0;

                        for(int r=0; r<recursos.size(); r++){

                                String recursiAux =
recursos.get(r);
                                //En esta parte Cambiar la
obtencion de getI

   if(pnali.getR().compareTo(recursiAux)==0){
                                        idenRecursis = r;
                                }
                        }//Termina primer elemento del indice del
conjunto i
```

```
                    int idenPer=0;
                    for(int t=0; t<periodos.size(); t++){

                            String TAux = periodos.get(t);

                    //En esta parte Cambiar la obtencion de
getI

   if(pnali.getT().compareTo(TAux)==0){
                                    idenPer = t;
                            }
                    }//Termina primer elemento del indice del
conjunto t

   pEN1[idenRecursis][idenPer]=pnali.getPNmasrt();
            }//Fin lectura de Penalidad 1
            //*****************************************


            //Lectura Penalidad 2
            //*****************************************
            pEN2=new double[numR][numT];//Penalidad Extra
ArrayList<PenalidadMENOS> penalidadmenos =
(ArrayList)datos.getPenalidadesMenos();
            for(int i=0;i<penalidadmenos.size();i++){

                    //Se crea un vector demanda auxiliar

                    PenalidadMENOS pnalim =
penalidadmenos.get(i);
                    //Inicio Lectura primer elemento del
conjunto P
                    int idenRecursis=0;

                    for(int r=0; r<recursos.size(); r++){

                            String recursiAux =
recursos.get(r);
            //En esta parte Cambiar la obtencion de getI

   if(pnalim.getR().compareTo(recursiAux)==0){
                                    idenRecursis = r;
                            }
                    }//Termina primer elemento del indice del
conjunto i
                    int idenPer=0;
                    for(int t=0; t<periodos.size(); t++){

                            String TAux = periodos.get(t);

            //En esta parte Cambiar la obtencion de getI

   if(pnalim.getT().compareTo(TAux)==0){
                                    idenPer = t;
                            }
                    }//Termina primer elemento del indice del
conjunto t

   pEN2[idenRecursis][idenPer]=pnalim.getPNmenosrt();
            }//Fin lectura de Penalidad 2
            //*****************************************
```

```
            //Lectura rBom
            //*****************************************
            nik=new double[numI][numK];//rBOM
            ArrayList<RBOM> NIKa = (ArrayList)datos.getRBOMs();

            for(int i=0;i<NIKa.size();i++){

                    //Se crea un vector demanda auxiliar

                    RBOM nikas = NIKa.get(i);

                    //Inicio Lectura primer elemento del
conjunto P
                    int idenSKU=0;
                    for(int ii=0; ii<skuses.size(); ii++){

                            String SKUAux = skuses.get(ii);

                    //En esta parte Cambiar la obtencion de
getI

   if(nikas.getI().compareTo(SKUAux)==0){
                                    idenSKU = ii;
                            }
                    }//Termina primer elemento del indice del
conjunto i
                    int idenStroke=0;
                    for(int k=0; k<strokes.size(); k++){

                            String StrokeAux =
strokes.get(k);
                    //En esta parte Cambiar la obtencion de
getI

   if(nikas.getK().compareTo(StrokeAux)==0){
                                    idenStroke = k;
                            }
                    }//Termina primer elemento del indice del
conjunto i
                    nik[idenSKU][idenStroke]=nikas.getNik();
            }//Fin lectura de Penalidad 2
            //*****************************************


            //Lectura Bom
            //*****************************************
            mik=new double[numI][numK];//BOM
            ArrayList<BOM> MIKa = (ArrayList)datos.getBOMs();

            for(int i=0;i<MIKa.size();i++){

                    //Se crea un vector demanda auxiliar

                    BOM mikas = MIKa.get(i);

                    //Inicio Lectura primer elemento del
conjunto P
                    int idenSKU=0;
```

```
                    for(int ii=0; ii<skuses.size(); ii++){

                            String SKUAux = skuses.get(ii);

                            //En esta parte Cambiar la
obtencion de getI

   if(mikas.getI().compareTo(SKUAux)==0){
                                    idenSKU = ii;
                            }
                    }//Termina primer elemento del indice del
conjunto i
                    int idenStroke=0;
                    for(int k=0; k<strokes.size(); k++){

                            String StrokeAux =
strokes.get(k);
                            //En esta parte Cambiar la
obtencion de getI

   if(mikas.getK().compareTo(StrokeAux)==0){
                                    idenStroke = k;
                            }
                    }//Termina primer elemento del indice del
conjunto i
                    mik[idenSKU][idenStroke]=mikas.getMik();
            }//Fin lectura de Penalidad 2
            //*****************************************

            //Lectura Coeficientes recursos de las ecuaciones
            //*****************************************
            rEkr=new double[numK][numR];//Coeficientes recursos
ArrayList<CoefTecno> coefi = (ArrayList)datos.getCoeRecurs();

            for(int i=0;i<coefi.size();i++){

                    //Se crea un vector demanda auxiliar

                    CoefTecno tec = coefi.get(i);

                    //Inicio Lectura primer elemento del
conjunto P
                    int idenStroke=0;
                    for(int k=0; k<strokes.size(); k++){

                            String StrokeAux =
strokes.get(k);
                            //En esta parte Cambiar la
obtencion de getI

   if(tec.getK().compareTo(StrokeAux)==0){
                                    idenStroke = k;
                            }
                    }//Termina primer elemento del indice del
conjunto k
                    int idenRecursis=0;

                    for(int r=0; r<recursos.size(); r++){

                            String recursiAux =
recursos.get(r);
```

```
  //En esta parte Cambiar la obtencion de getI

   if(tec.getR().compareTo(recursiAux)==0){
                                        idenRecursis = r;
                                }
                        }//Termina primer elemento del indice del
conjunto i

   rEkr[idenStroke][idenRecursis]=tec.getRErk();
             }//Fin lectura de coeficiente recursos de las
ecuaciones
             //*****************************************



             //Lectura Precios de Venta
             //*****************************************
             pVi=new double[numI];//Precio de venta de producto
             ArrayList<PrecioProd> pre =
(ArrayList)datos.getPrecios();
             for(int i=0;i<pre.size();i++){

                      //Se crea un vector demanda auxiliar

                      PrecioProd prec = pre.get(i);

                      int idenSKU=0;
                      for(int ii=0; ii<skuses.size(); ii++){

                               String SKUAux = skuses.get(ii);


   if(prec.getI().compareTo(SKUAux)==0){
                                        idenSKU = ii;
                                }
                      }//Termina elemento

                      pVi[idenSKU]=prec.getPVi();
             }//Fin lectura de precios de venta
             //*********************************************


             //Lectura Inventarios iniciales
             //*****************************************
             iNI=new double[numI];//Precio de venta de producto
ArrayList<InventarioIncial> invinii =
(ArrayList)datos.getInvInicial();
             for(int i=0;i<invinii.size();i++){

                      //Se crea un vector demanda auxiliar

                      InventarioIncial inicial = invinii.get(i);

                      int idenSKU=0;
                      for(int ii=0; ii<skuses.size(); ii++){

                               String SKUAux = skuses.get(ii);

                               //En esta parte se obtiene getI

   if(inicial.getI().compareTo(SKUAux)==0){
```

```
                                        idenSKU = ii;
                                }
                        }//Termina elemento

                        iNI[idenSKU]=inicial.getINI();
                }//Fin lectura de precios de venta
                //*********************************************

                cHit=new double[numI][numT];
ArrayList<CosteInventory> Chin =
(ArrayList)datos.getCosteAlmacena();
                for(int i=0;i<Chin.size();i++){

                        //Se crea un vector demanda auxiliar

                        CosteInventory aCH = Chin.get(i);

                        //Inicio Lectura primer elemento del
conjunto P
                        int idenSKU=0;
                        for(int ii=0; ii<skuses.size(); ii++){

                                String SKUAux = skuses.get(ii);

                                //En esta parte Cambiar la
obtencion de getI

   if(aCH.getI().compareTo(SKUAux)==0){
                                        idenSKU = ii;
                                }
                        }//Termina primer elemento del indice del
conjunto i
                        int idenPer=0;
                        for(int t=0; t<periodos.size(); t++){

                                String TAux = periodos.get(t);

                                //En esta parte Cambia la
obtencion de getI

   if(aCH.getT().compareTo(TAux)==0){
                                        idenPer = t;
                                }
                        }//Termina primer elemento del indice del
conjunto t
cHit[idenSKU][idenPer]=aCH.getCH();//Se captura el valor del
coste
                }//Fin lectura de activacion de compras

//*****************************************


                //Escritura de Bakorders
                cBit=new double[numI][numT];
ArrayList<CosteBackOrder> Bak =
(ArrayList)datos.getCosBackorders();
                for(int i=0;i<Chin.size();i++){

                        //Se crea un vector demanda auxiliar
```

```
                    CosteBackOrder vak = Bak.get(i);

                    //Inicio Lectura primer elemento del
conjunto P
                    int idenSKU=0;
                    for(int ii=0; ii<skuses.size(); ii++){

                            String SKUAux = skuses.get(ii);

                            //En esta parte Cambia la
obtencion de getI

   if(vak.getI().compareTo(SKUAux)==0){
                                    idenSKU = ii;
                            }
                    }//Termina primer elemento del indice del
conjunto i
                    int idenPer=0;
                    for(int t=0; t<periodos.size(); t++){

                            String TAux = periodos.get(t);

                            //En esta parte Cambiar la
obtencion de getI

   if(vak.getT().compareTo(TAux)==0){
                                    idenPer = t;
                            }
                    }//Termina primer elemento del indice del
conjunto t
cBit[idenSKU][idenPer]=vak.getCBit();//Se captura el valor del
coste
            }//Fin lectura de BackOrders
            //*****************************************


            //Lectura LeadTime de cada Procedente de cada
Stroke
            //*****************************************
            lt=new int[numK]; //Lead time de productos
            ArrayList<Leadtime> ltS =
(ArrayList)datos.getLeadtimes();
            for(int i=0;i<ltS.size();i++){

                    //Se crea un vector demanda auxiliar

                    Leadtime LT = ltS.get(i);

                    int idenStroke=0;
                    for(int k=0; k<strokes.size(); k++){

                            String StrokeAux =
strokes.get(k);
                            //En esta parte Cambiar la
obtencion de getI

   if(LT.getK().compareTo(StrokeAux)==0){
                                    idenStroke = k;
                            }
                    }//Termina primer elemento del indice del
conjunto k
```

```
                    lt[idenStroke]=LT.getLTk();
                }//Fin lectura de precios de venta
                //*******************************************



                }catch(Exception eP){
                    eP.printStackTrace();
                }
                }//Fin del método de leer el XML

    // Declaración de los métodos asociados
    public LeerXML(){
    //Este es el constructor de la clase
                super();
    }

    public int getTao() {
                return tao;
                }

                public void setTao(int tao) {
                this.tao = tao;
                }

                public int getLa() {
                return La;
}

                public static void setLa(int la) {
                La = la;
                }

                public static int getLb() {
                return Lb;
                }

                public static void setLb(int lb) {
                Lb = lb;
                }
}
```

Ocurrencias del modelo Fase II

```
package EstocaDET;

import EstocaDET.*;

public class Ocurrencias {

    LeerXML le=new LeerXML();

    public double[][] invtis;
    public double backorderis[][];
    public double [][]costrokes2;
    public double [][]costrokes;
    public double [][]penrecursis1;
    public double [][]penrecursis2;
    public double [] falta;
    public double [] stock;
    public double []benesperado2;
```

```
   public double [][] production2;
   public double [][] Stroke2;
   public double [][] compras2;
   public double [] ventas2;
   public double []VR2;
   public double Ft1;
   public double Ft2;
   public double [][]ventasRit;
   public double []ventas;


//public double fcostes[];//En cada posición va a contener las
funciones objetivas


   public Ocurrencias(int TAO){

            invtis=new double[le.numI][TAO];
            backorderis=new double[le.numI][TAO];
            penrecursis1 = new double[le.numR][le.numT-TAO];
            penrecursis2 = new double[le.numR][le.numT-TAO];
            costrokes2 = new double[le.numK][le.numT-TAO];
            costrokes = new double[le.numK][le.numT];
            falta =new double [(le.numT -TAO)];
            stock =new double [le.numT - TAO];
            benesperado2=new double [le.numT - TAO];
            Stroke2 = new double[le.numK][le.numT-TAO];
            compras2 =new double[le.numI][le.numT-TAO];
            ventasRit=new double[le.numI][TAO];
            ventas = new double [TAO];

            VR2 =new double[le.numT-TAO];
            Ft1=0;


   //fcostes =new double [le.numT-TAO];
   }

public void simular (LeerXML le, SolveM1 sol1, int iterocur,
int TAO, Resultados rez){


   //Sol1 es el objeto del que se va a extraer la producción

//EL iteroccur se utiliza para acceder a la ocurrencia
respectiva en LeerXML

            String nombreOk=
"entradas/PROBLEMA("+(iterocur+1)+").txt"; //Nombre de la
instancia a resolver
            System.out.println("Este es el "+nombreOk);
            System.out.println("----------------------------
------");
            double valor[][]=new double[le.numI][TAO];

            for (int i=0; i<le.numI;i++){
                  for(int t=0;t<(TAO);t++){
                        //Cálculo del inventario y
backorder
```

```
if(t>0){//Cálculo para un periodos posteriores al periodo
inicial

valor[i][t]=invtis[i][t-1]-backorderis[i][t-
1]+sol1.produccion1[i][t][0]+sol1.compras1[i][t][0]-
le.dEMito[i][t][iterocur]-sol1.consumidos[i][t][0];
            }else{//Cálculo de ventas para el periodo inicial


valor[i][t]=le.iNI[i]+sol1.produccion1[i][t][0]+sol1.compras1[i
][t][0]-le.dEMito[i][t][iterocur]-sol1.consumidos[i][t][0];
                        }
                        //Cálculo de las ventas reales del sistema
                        if(valor[i][t]>=0){
                                invtis[i][t]=valor[i][t];
                                backorderis[i][t]=0;
                        }else{
                                invtis[i][t]=0 ;

   backorderis[i][t]=Math.abs(valor[i][t]);
                        }
                        //Cálculo de las ventas reales
                        if(t>0){//Las ventas para despues del
primer periodo
ventasRit[i][t]=(Math.max(0, le.dEMito[i][t][iterocur]-
backorderis[i][t]+backorderis[i][t-1]));

                        }else{//Ventas para el periodo inicial
ventasRit[i][t]=(Math.max(0, le.dEMito[i][t][iterocur]-
backorderis[i][t]));
                                }
//System.out.println("en i= "+i+" t = "+(t+1)+" Ventas de " +
ventasRit[i][t] + " y Back de  "+ backorderis[i][t]);
            } //End Periodos
            }//End SKU


            //Cálculo de las ventas reales por periodo de
tiempo
            for(int t=0; t<(TAO);t++){
                        double acumm=0;
                        for(int i=0;i<le.numI;i++){

   acumm=acumm+ventasRit[i][t]*le.pVi[i];
                        }
                        ventas[t]=acumm;
                        //System.out.println("LAs ventas son:
"+ventas[t]);
            }

            //Cálculo del las Backorder acumuladas el sistema
            for(int i=0;i<le.numI;i++){
                        for(int t=0;t<(TAO);t++){
                        Ft1=Ft1+backorderis[i][t];
                        }
            }

            //Cálculo de las
            double[] INventariofin=new double[le.numI];
            double[] BAKIRfin=new double[le.numI];
```

```
            //Se escriben los inventarios finales

   //System.out.println("TAO es "+ TAO);
   for(int i=0;i<le.numI;i++){
            INventariofin[i]=invtis[i][TAO-1];
//System.out.println("El inventario final es
"+INventariofin[i]);

            BAKIRfin[i]=backorderis[i][TAO-1];
//System.out.println("La Faltante final es "+BAKIRfin[i]);
   }


//Se escribe el los datos para cada uno de los modelos
seleccionados
EscribirTXT2 txt2 =new EscribirTXT2(le, sol1, INventariofin,
BAKIRfin, nombreOk, iterocur);
SolveM2 resolver2 = new SolveM2();//Se crea el objeto para
resolver el problema


   try {
            resolver2.solveM2(le, nombreOk);

            costrokes2=resolver2.costrokes2;
            penrecursis1=resolver2.getPenrecursis1();
            penrecursis2=resolver2.getPenrecursis2();
            falta=resolver2.getFalta();
            stock=resolver2.getStock();
            costrokes2=resolver2.getCostrokes2();
            production2=resolver2.getProduccion2();
            Stroke2=resolver2.getStroke2();
            compras2=resolver2.getCompras2();
            ventas2=resolver2.getVentas2();
            VR2=resolver2.getVR();
            Ft2=resolver2.getFaltTot();

   } catch (Exception e) {

            System.out.println("Error al resolver");

            e.printStackTrace();
   }
   } //FIn de la clase simular


   public double[][] getInvtis() {
   return invtis;
   }

   public double[][] getBackorderis() {
            return backorderis;
   }

   public double[][] getCostrokes2() {
            return costrokes2;
   }

   public double[][] getPenrecursis1() {
            return penrecursis1;
   }
```

```
    public double[][] getPenrecursis2() {
            return penrecursis2;
    }

    public double[] getFalta() {
            return falta;
    }

    public double[] getStock() {
            return stock;
    }

    public double[] getBenesperado2() {
            return benesperado2;
    }
}
```

Clase para escribir resultados

```
package EstocaDET;

import java.io.File;

import LectorXML.CosteInventory;

import jxl.write.Number;
import jxl.Workbook;
import jxl.write.Label;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;

public class Resultados {


    //Etapa 1
    public double [][] costrokes;
    public double [][] CosRecursosGen;
    public double [] benesperado;
    public double backorderis[][][];
    public double produceM1[][][];
    public double[][] strokes1etapa;
    public double [][][]compras2etapa;
    public double [][][]compras1etapa;
    public double invtis2[][][];
    public double ventis1[][];
    public double ventis2[][];
    //Etapa 2
    public double [][][] costrokes2;
    public double [][][] strokes2etapa;
    public double [][][] produceM2;
    public double [][][] penrecursis1;
    public double [][][] penrecursis2;
    public double [][] falta;
    public double [][] stock;
    public double [][]          CstrokeGen;
    public double [] CInventarioGen;
    public double [] CBackOrderGen;
    public double [][] CRPEN1Gen;
    public double [][] CRPEN2Gen;
    public double [][] ProdGen;
```

```
    public double [][] StrokeGen;
    public double [][] ComprasGen;
    public double [][] CostCompraGen;
    public double [] VentasGen;
    public double [] CostTotGen;
    public double [][] VResidual;
    public double [] VRGEN;
    public double NV=0;
    public double ft2[];
    public double ft1[];
    public double faltanteTot;


    public Resultados(LeerXML le, int tao) {

    costrokes = new double [le.numK][le.numT];
    invtis2 = new double [le.numI][tao][le.numO];
    backorderis = new double [le.numI][tao][le.numO];

    penrecursis1 = new double[le.numR][le.numT-tao][le.numO];
    penrecursis2 = new double[le.numR][le.numT-tao][le.numO];
    costrokes2 = new double[le.numK][le.numT-tao][le.numO];

    falta =new double [(le.numT -tao)][le.numO];
    stock =new double [le.numT - tao][le.numO];
    benesperado=new double [le.numT ];
    produceM1=new double [le.numI][le.numT][le.numE];
    produceM2=new double [le.numI][le.numT-tao][le.numO];

    strokes2etapa= new double[le.numK][le.numT-tao][le.numO];

    strokes1etapa= new double[le.numK][le.numT];

    compras2etapa= new double[le.numI][le.numT-tao][le.numO];

    compras1etapa= new double[le.numI][le.numT][le.numE];

    VResidual =  new double[le.numT-tao][le.numO];
    ventis2 = new double[le.numT-tao][le.numO];
    ventis1 =new double[le.numT][le.numO];
    CostTotGen = new double[le.numT];
    VRGEN = new double[le.numT];
    ft2= new double[le.numO];
    ft1= new double[le.numO];

    //Resultados Generales
CstrokeGen = new double [le.numK][le.numT]; //Costes de Stroke
promedios
CInventarioGen=new double [le.numT]; // Costes promedios de
inventario
CBackOrderGen=new double [le.numT];     // Costes promedios de
BackOrder
CRPEN1Gen=new double[le.numR][le.numT]; // Costes por sobreuso
de la capacidad
CRPEN2Gen=new double[le.numR][le.numT]; //Costes por subuso de
la capacidad
CosRecursosGen = new double[le.numR][le.numT]; //Costes de
reserva de la capacidad
CostCompraGen = new double[le.numI][le.numT]; //Coste de
materia prima
VentasGen = new double[le.numT]; //Ventas promedios totales
```

```
    //Valor residual del inventario
    ProdGen=new double[le.numI][le.numT];
    StrokeGen=new double[le.numK][le.numT];
    ComprasGen = new double[le.numI][le.numT];
    }

    public void Resul(LeerXML le,SolveM1 m1){
              costrokes=m1.costrokes;
              CosRecursosGen=m1.cosrecursis;
              produceM1=m1.produccion1;
              strokes1etapa=m1.Stroke1;
              compras1etapa=m1.compras1;


    } //Fin result

public void Resul2(LeerXML le,Ocurrencias occ, int iterocur,
int tao){

              for(int t=0;t<(le.numT-tao);t++){
              VResidual[t][iterocur]=occ.VR2[t];
              }

              ft2[iterocur]=occ.Ft2;
              ft1[iterocur]=occ.Ft1;


              //Se extrae las ventas de la segunda etapa en el
xls

              for(int t=0;t<(le.numT-tao);t++){

              ventis2[t][iterocur]=occ.ventas2[t];

              }


              //Se extrae la produccion para colocarla en el xls
              for(int i=0;i<le.numI;i++){
                        for(int t=0;t<(le.numT-tao);t++){


    produceM2[i][t][iterocur]=occ.production2[i][t];
                        }
              }

              //Se extrae las compras para colocarla en el xls
              for(int i=0;i<le.numI;i++){
                        for(int t=0;t<(le.numT-tao);t++){


    compras2etapa[i][t][iterocur]=occ.compras2[i][t];

                        }
              }

              for(int i=0;i<le.numI;i++){
                        for(int t=0;t<tao;t++){
```

```
   invtis2[i][t][iterocur]=occ.invtis[i][t];


   backorderis[i][t][iterocur]=occ.backorderis[i][t];
//System.out.println("el BackOrder es
"+backorderis[i][t][iterocur]);
                        }
                }

                for(int r=0;r<le.numR;r++){
                        for(int t=0;t<le.numT-tao;t++){


   penrecursis1[r][t][iterocur]=occ.penrecursis1[r][t];
                        }
                }

                for(int r=0;r<le.numR;r++){
                        for(int t=0;t<le.numT-tao;t++){


   penrecursis2[r][t][iterocur]=occ.penrecursis2[r][t];
                                }
                }
                //Calculando las ventas reales del sistema
                for(int t=0;t<tao;t++){

                        ventis1[t][iterocur]=occ.ventas[t];
                }

                for(int t=0;t<le.numT-tao;t++){

                        falta[t][iterocur]=occ.falta[t];
                }

                for(int t=0;t<le.numT-tao;t++){

                        stock[t][iterocur]=occ.stock[t];

                }

                for(int k=0;k<le.numK;k++){
                        for(int t=0;t<(le.numT-tao);t++){

   costrokes2[k][t][iterocur]=occ.costrokes2[k][t];
                        }
                }

                for(int k=0;k<le.numK;k++){
                        for(int t=0;t<(le.numT-tao);t++){

   strokes2etapa[k][t][iterocur]=occ.Stroke2[k][t];
                        }
                }

   }//Fin Result2


   public void ResGeneral(LeerXML le, int tao){
```

```
                //Relleno faltantes total
                double acumm=0;
                //double [] acumm2=new double[le.numO];
                for(int o=0;o<le.numO;o++){
                          acumm=acumm + (ft1[o]+ft2[o]);

                }
                faltanteTot=(acumm/le.numO);

                //Cálculo la demanda total
                for(int o=0;o<le.numO;o++){
                          for(int i=0;i<le.numI;i++){
                                    for(int t=0;t<le.numT;t++){


   acumm=acumm+le.dEMito[i][t][o];
                                    }
                          }
                }
                NV=1-faltanteTot/(acumm/le.numO);

                //Relleno del valor residual

                for(int t=0;t<le.numT;t++){
                          //System.out.println(" es  "+le.numT);
                          if(t>=tao){

                                    double acum=0;
                                    for(int o=0;o<le.numO;o++){

                                              acum=acum +
(VResidual[t-tao][o]);
                                    }
                                    VRGEN[t]=(acum/le.numO);

                          }
                }

                //Rellenado de la matriz de ventas

                for(int t=0;t<le.numT;t++){
                          //System.out.println(" es  "+le.numT);
                          if(t>=tao){

                                    double acum=0;
                                    for(int o=0;o<le.numO;o++){

                                              acum=acum + (ventis2[t-
tao][o]);
// Aquí van las ventas reales calculadas de la simulacion

                                    }
                                    VentasGen[t]=(acum/le.numO);

                          }else{
                                    double acum=0;
                                    for(int o=0;o<le.numO;o++){


   //System.out.println(produceM1[i][t][0]);
```

```
                                        acum=acum
+ventis1[t][o];
                                }
                                VentasGen[t]=(acum/le.numO);

                                }
                        //System.out.println(VentasGen[t]);
                }


                //Rellenado de la matriz de la produccion
                for(int i=0;i<le.numI;i++){
                        for(int t=0;t<le.numT;t++){

                                //System.out.println(" es
"+le.numT);
                                if(t>=tao){

                                        double acum=0;
                                        for(int
o=0;o<le.numO;o++){
                                        acum=acum +
(produceM2[i][t-tao][o]);

                                        }

   ProdGen[i][t]=(acum/le.numO);

                                }else{

   //System.out.println(produceM1[i][t][0]);

   ProdGen[i][t]=produceM1[i][t][0];
                                }
                        }
                }

                //Rellenado de la matriz de compras
                for(int i=0;i<le.numI;i++){

   for(int t=0;t<le.numT;t++){
                //System.out.println(" es  "+le.numT);
                if(t>=tao){

                        double acum=0;
                        for(int o=0;o<le.numO;o++){

                        acum=acum + (compras2etapa[i][t-tao][o]);

                        }
                        ComprasGen[i][t]=(acum/le.numO);

                        }else{

   ComprasGen[i][t]=compras1etapa[i][t][0];
                        }
                }
   }

   //Rellenado de la matriz General de Strokes
   for(int k=0;k<le.numK;k++){
```

```
            for(int t=0;t<le.numT;t++){
                    //System.out.println(" es  "+le.numT);
                    if(t>=tao){

                            double acum=0;
                    for(int o=0;o<le.numO;o++){

                    acum=acum + (strokes2etapa[k][t-tao][o]);

                    }
                    StrokeGen [k][t]=(acum/le.numO);

                    }else{

                            StrokeGen
[k][t]=strokes1etapa[k][t];
                    }
            }
   }

//Rellenado de la matriz General de costes de Strokes
   for(int k=0;k<le.numK;k++){

            for(int t=0;t<le.numT;t++){
                    //System.out.println(" es  "+le.numT);
                    if(t>=tao){

                    double acum=0;
                    for(int o=0;o<le.numO;o++){

                    acum=acum + (costrokes2[k][t-tao][o]);

                    }
                    CstrokeGen [k][t]=(acum/le.numO);

                    }else{

                            CstrokeGen
[k][t]=costrokes[k][t];
                    }
            }
   }

//Rellenado de la matriz General de costes de Penalidad por
subuso de los recursos
            for(int r=0;r<le.numR;r++){
                    for(int t=0;t<le.numT;t++){

                    if(t>=tao){
                            double acum=0;
                            for(int o=0;o<le.numO;o++){

                            acum=acum + penrecursis1[r][t-
tao][o];                                                }

                            CRPEN1Gen[r][t]=acum/le.numO;
                            }else{

                    CRPEN1Gen[r][t]=0;
//Porque se fabrica exactamente la capacidad con los stroke
                    }
```

```
                    }
                }

//Rellenado de la matriz General de costes de Penalidad por
sobreuso de los recursos
                for(int r=0;r<le.numR;r++){
                        for(int t=0;t<le.numT;t++){

                        if(t>=tao){
                        double acum=0;
                        for(int o=0;o<le.numO;o++){

                        acum=acum + penrecursis2[r][t-tao][o];

                        }
                        CRPEN2Gen [r][t]=acum/le.numO;
                        }else{

                        CRPEN2Gen[r][t]=0;
//Porque se fabrica exactamente la capacidad con los stroke
                                }

                        }
                }

   //Rellenado y cálculo de los costes de compra de materiales
                for(int i=0;i<le.numI;i++){
                        for(int t=0;t<le.numT;t++){



   CostCompraGen[i][t]=ComprasGen[i][t]*le.pCit[i][t];
                        }
                }

   //Relleno de matriz de costes de inventario

                double [][] valor=new double [tao][le.numO];
                for(int o=0;o<le.numO;o++){

                        for(int t=0;t<tao;t++){
                        for(int i=0;i<le.numI;i++){


   valor[t][o]=valor[t][o]+le.cHit[i][t]*invtis2[i][t][o];

                        }

                        }
                }

                for(int t=0;t<le.numT;t++){

//System.out.println("jejejejejjee "+tao);
                        if(t>=tao){
                        double acum=0;
                        for(int o=0;o<le.numO;o++){

                        acum=acum+stock[t-tao][o];
```

```
                        }

                        CInventarioGen[t]=acum/le.numO;

                        }else{
                        double acum=0;
                        for(int o=0;o<le.numO;o++){

                        acum=acum +valor[t][o];

                        }

                        CInventarioGen[t]=acum/le.numO;
                }
        }

//Rellenando los costes de backorder totales

        for(int t=0;t<le.numT;t++){

                if(t>=tao){
                        double acum=0;

                        for(int o=0;o<le.numO;o++){

                                acum=acum+falta[t-tao][o];

                        }

                        CBackOrderGen[t]=acum/le.numO;
//System.out.println("El coste promedio es
"+CBackOrderGen[t]);
                }else{
                        double acum=0;
                        for(int o=0;o<le.numO;o++){

                                for(int i=0;i<le.numI;i++){



        acum=acum+le.cBit[i][t]*backorderis[i][t][o];
                                }

                        }

                        CBackOrderGen[t]=acum/le.numO;
                }

        }//Fin for Backorders

//Cálculo del coste total por perido
        for(int t=0;t<le.numT;t++){
                double Cstroke=0;
                double Crecursos=0;
                double Ccompras=0;
                double CInBak=0;

                for(int i=0;i<le.numI;i++){
                        Ccompras=Ccompras+CostCompraGen[i][t];
                }
//Costes de Strokes
```

```
            for(int k=0;k<le.numK;k++){
                    Cstroke=Cstroke+CstrokeGen[k][t];

            }
//Coste de recursos
            for(int r=0;r<le.numR;r++){
Crecursos=Crecursos+CRPEN1Gen[r][t]+CRPEN2Gen[r][t]+CosRecursos
Gen[r][t];
            }
   //Costes de inventario y Backorder
                    CInBak=CInventarioGen[t]+CBackOrderGen[t];
                    CostTotGen[t]=
CInBak+Cstroke+Crecursos+Ccompras;
   //System.out.println("SStroke "+ CostTotGen[t]);
            }

   //Cálculo del Beneficio
            for(int t=0;t<le.numT;t++){
                    benesperado[t]=VentasGen[t]+VRGEN[t]-
CostTotGen[t];
            }

   public void Escribir(LeerXML le, String nombre){

        try{
// Aquí se comienza a escribir el constructor de la información
WritableWorkbook libro = Workbook.createWorkbook(new
File(nombre));
                // Se crea un nuevo libro
//Se entra en una iteración que depende del número de
escenarios

//-----------------------------------------------------------------
--
WritableSheet hoja ;
   hoja = libro.createSheet("P & G", 0);
Label label;
//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(4,2,"Ventas totales" ); // Se crea el Label
hoja.addCell(label);// Se escribe el Label en la celda
label = new Label(5,2,"Valor Residual del inv" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
label = new Label(6,2,"Costes Totales" ); // Se crea el Label
hoja.addCell(label);// Se escribe el Label en la celda
label = new Label(7,2,"Beneficio Esperado" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
// Costes de Strokes
for(int t=0; t<le.numT; t++){
    Number number =new Number(3,3+t, (t+1));
hoja.addCell(number);
}
//Escribir los Costes de Stroke por periodo

   for(int t=0; t<le.numT; t++){
   Number number =new Number(4,3+t, VentasGen[t]);
   hoja.addCell(number);
   number =new Number(5,3+t, VRGEN[t]);
   hoja.addCell(number);
```

```
   number =new Number(6,3+t, CostTotGen[t]);
   hoja.addCell(number);
   number =new Number(7,3+t, benesperado[t]);
   hoja.addCell(number);

}
label = new Label(0,0,"El nivel de servicio es: " ); // Se crea
el Label
hoja.addCell(label);// Se escribe el Label en la celda
Number number2 =new Number(2,0, NV);
hoja.addCell(number2);
//-------------------------------------------------------------
-----------
            hoja = libro.createSheet("Strokes & SKUs", 1);
//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(0,0,"Costes Promedio de Strokes por Peridos"
); // Se crea el Label
hoja.addCell(label);// Se escribe el Label en la celda
            // Costes de Strokes
                    for(int t=0; t<le.numT; t++){
                      Number number =new Number(t+4,2, (t+1));
                     hoja.addCell(number);
                }
      //Escribir los Costes de Stroke por periodo
              for(int k=0; k<le.numK; k++){

                    label = new Label(3,3+k, le.getStroke[k]);
                hoja.addCell(label);

                    for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+k, CstrokeGen[k][t]);
                hoja.addCell(number);
                    }
              }

      label = new Label(0,5+le.numK,"Strokes Promedios
Ejecutados por periodos" ); // Se crea el Label
hoja.addCell(label);// Se escribe el Label en la celda

              for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,2+4+le.numK, (t+1));
                hoja.addCell(number);
            }
      //Escribir los Stroke por periodo
             for(int k=0; k<le.numK; k++){

label = new Label(3,3+k+4+le.numK, le.getStroke[k]);
                hoja.addCell(label);

                    for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+k+4+le.numK,
(StrokeGen[k][t]));
                     hoja.addCell(number);
                     }
            }

        label = new Label(0,9+2*le.numK,"Generación Promedio de
Productos por periodos" ); // Se crea el Label
hoja.addCell(label);// Se escribe el Label en la celda
```

```
       //Escribir los Stroke por periodo
             for(int i=0; i<le.numI; i++){

label = new Label(3,3+i+4+2*le.numK+4, le.getSKU[i]);
                 hoja.addCell(label);

                     for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+i+4+2*le.numK+4,
(ProdGen[i][t]));

            hoja.addCell(number);
                     }
            }

            label = new Label(0,9+2*le.numK + le.numI
+4,"Compras Promedio de Materia Prima por periodos" ); // Se
crea el Label
hoja.addCell(label);// Se escribe el Label en la celda

   //Escribir los Stroke por periodo
                     for(int i=0; i<le.numI; i++){

label = new Label(3,3+i+4+2*le.numK+4+le.numI+4, le.getSKU[i]);
hoja.addCell(label);
                     for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+i+4+2*le.numK+4+le.numI+4,
(ComprasGen[i][t]));
//System.out.println("El valor es "+le.cOkt[k][t]+" es
"+CstrokeGen[k][t]/001);
hoja.addCell(number);
                     }
                     }
   label = new Label(0,3+4+2*le.numK+4+2*le.numI+6,"Costes de
Compras Promedio por periodos" );
hoja.addCell(label);
//Escribir los costes promedios de compras
for(int i=0; i<le.numI; i++){
label = new Label(3,3+i+4+2*le.numK+4+2*le.numI+8,
le.getSKU[i]);
hoja.addCell(label);
            for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+i+4+2*le.numK+4+2*le.numI+8,
(CostCompraGen[i][t]));
//System.out.println("El valor es "+le.cOkt[k][t]+" es
"+CstrokeGen[k][t]/001);
                     hoja.addCell(number);
                     }
}

//-----------------------------------------------------------------
---

hoja = libro.createSheet("Coste de los recursos", 2);
//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(0,0,"Costes de Recursos" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
// Costes de Recursos
for(int t=0; t<le.numT; t++){
   Number number =new Number(t+4,2, (t+1));
```

```
hoja.addCell(number);
}
//Escribir Los inventarios totales por periodo
for(int r=0; r<le.numR; r++){
   label = new Label(3,3+r, le.getRecursos[r]);
hoja.addCell(label);
for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+r, CosRecursosGen[r][t]);
hoja.addCell(number);
}
}
//-------------------------------------------------------------
---

hoja = libro.createSheet("Coste Inventario y Faltantes", 3);
//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(4,2,"Costes de Inventario" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
label = new Label(5,2,"Costes de Faltantes" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
// Costes de Strokes
for(int t=0; t<le.numT; t++){
Number number =new Number(3,3+t, (t+1));
hoja.addCell(number);
}
//Escribir Los Costes de Stroke por periodo


for(int t=0; t<le.numT; t++){
   Number number =new Number(4,3+t, CInventarioGen[t]);
   hoja.addCell(number);
   number =new Number(5,3+t, CBackOrderGen[t]);
   hoja.addCell(number);
}
//-------------------------------------------------------------
---

hoja = libro.createSheet("Penalidad Sobre uso", 4);
//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(0,0,"Penalidad Sobre Uso"); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
// Costes de Recursos
for(int t=0; t<le.numT; t++){
   Number number =new Number(t+4,2, (t+1));
   hoja.addCell(number);
}

//Escribir Los inventarios totales por periodo
for(int r=0; r<le.numR; r++){

   label = new Label(3,3+r, le.getRecursos[r]);
   hoja.addCell(label);

   for(int t=0; t<le.numT; t++){

   Number number =new Number(t+4,3+r, CRPEN1Gen[r][t]);
```

```
   hoja.addCell(number);
   }
}
//-------------------------------------------------------------
---
hoja = libro.createSheet("Penalidad Sub uso", 5);

//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(0,0,"Penalidad Sub Uso" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda

// Costes de Recursos
for(int t=0; t<le.numT; t++){
   Number number =new Number(t+4,2, (t+1));
   hoja.addCell(number);
}

//Escribir Los inventarios totales por periodo
for(int r=0; r<le.numR; r++){

   label = new Label(3,3+r, le.getRecursos[r]);
   hoja.addCell(label);

   for(int t=0; t<le.numT; t++){
   Number number =new Number(t+4,3+r, CRPEN2Gen[r][t]);
   hoja.addCell(number);
   }
}
//-------------------------------------------------------------
---

//Aquí van los métodos de la hoja de cálculo resumen

//--------------------------
//Se escribe el libro
libro.write();
libro.close();

System.out.println("Escribio en Excel");

}catch(Exception eP){

eP.printStackTrace();
}//Fin Catch
}
}
```

Clase para resolver el primer modelo (Fase I)

```
package EstocaDET;

import java.io.File;

import LectorXML.CosteInventory;

import jxl.write.Number;
import jxl.Workbook;
import jxl.write.Label;
import jxl.write.WritableSheet;
```

```
import jxl.write.WritableWorkbook;

public class Resultados {


   //Etapa 1
   public double [][] costrokes;
   public double [][] CosRecursosGen;
   public double [] benesperado;
   public double backorderis[][][];
   public double produceM1[][][];
   public double[][] strokes1etapa;
   public double [][][]compras2etapa;
   public double [][][]compras1etapa;
   public double invtis2[][][];
   public double ventis1[][];
   public double ventis2[][];
   //Etapa 2
   public double [][][] costrokes2;
   public double [][][] strokes2etapa;
   public double [][][] produceM2;
   public double [][][] penrecursis1;
   public double [][][] penrecursis2;
   public double [][] falta;
   public double [][] stock;
   public double [][]         CstrokeGen;
   public double [] CInventarioGen;
   public double [] CBackOrderGen;
   public double [][] CRPEN1Gen;
   public double [][] CRPEN2Gen;
   public double [][] ProdGen;
   public double [][] StrokeGen;
   public double [][] ComprasGen;
   public double [][] CostCompraGen;
   public double [] VentasGen;
   public double [] CostTotGen;
   public double [][] VResidual;
   public double [] VRGEN;
   public double NV=0;
   public double ft2[];
   public double ft1[];
   public double faltanteTot;


   public Resultados(LeerXML le, int tao) {

   costrokes = new double [le.numK][le.numT];
   invtis2 = new double [le.numI][tao][le.numO];
   backorderis = new double [le.numI][tao][le.numO];

   penrecursis1 = new double[le.numR][le.numT-tao][le.numO];
   penrecursis2 = new double[le.numR][le.numT-tao][le.numO];
   costrokes2 = new double[le.numK][le.numT-tao][le.numO];

   falta =new double [(le.numT -tao)][le.numO];
   stock =new double [le.numT - tao][le.numO];
   benesperado=new double [le.numT ];
   produceM1=new double [le.numI][le.numT][le.numE];
   produceM2=new double [le.numI][le.numT-tao][le.numO];
```

```
   strokes2etapa= new double[le.numK][le.numT-tao][le.numO];

   strokes1etapa= new double[le.numK][le.numT];

            compras2etapa= new double[le.numI][le.numT-
tao][le.numO];
            compras1etapa= new
double[le.numI][le.numT][le.numE];
   VResidual =  new double[le.numT-tao][le.numO];
   ventis2 = new double[le.numT-tao][le.numO];
   ventis1 =new double[le.numT][le.numO];
   CostTotGen = new double[le.numT];
   VRGEN = new double[le.numT];
   ft2= new double[le.numO];
   ft1= new double[le.numO];

   //Resultados Generales
CstrokeGen = new double [le.numK][le.numT]; //Costes de Stroke
promedios
CInventarioGen=new double [le.numT]; // Costes promedios de
inventario
CBackOrderGen=new double [le.numT];     // Costes promedios de
BackOrder
CRPEN1Gen=new double[le.numR][le.numT]; // Costes por sobreuso
de la capacidad
CRPEN2Gen=new double[le.numR][le.numT]; //Costes por subuso de
la capacidad
CosRecursosGen = new double[le.numR][le.numT]; //Costes de
reserva de la capacidad
CostCompraGen = new double[le.numI][le.numT]; //Coste de
materia prima
   VentasGen = new double[le.numT]; //Ventas promedios totales

   //Valor residual del inventario
   ProdGen=new double[le.numI][le.numT];
   StrokeGen=new double[le.numK][le.numT];
   ComprasGen = new double[le.numI][le.numT];
}

   public void Resul(LeerXML le,SolveM1 m1){
   costrokes=m1.costrokes;
   CosRecursosGen=m1.cosrecursis;
   produceM1=m1.produccion1;
   strokes1etapa=m1.Stroke1;
   compras1etapa=m1.compras1;

} //Fin result

public void Resul2(LeerXML le,Ocurrencias occ, int iterocur,
int tao){

   for(int t=0;t<(le.numT-tao);t++){
            VResidual[t][iterocur]=occ.VR2[t];
   }

   ft2[iterocur]=occ.Ft2;
   ft1[iterocur]=occ.Ft1;

   //Se extrae las ventas de la segunda etapa en el xls
```

```
    for(int t=0;t<(le.numT-tao);t++){

              ventis2[t][iterocur]=occ.ventas2[t];

    }

    //Se extrae la produccion para colocarla en el xls
    for(int i=0;i<le.numI;i++){
              for(int t=0;t<(le.numT-tao);t++){

              produceM2[i][t][iterocur]=occ.production2[i][t];

              }
    }

    //Se extrae las compras para colocarla en el xls
    for(int i=0;i<le.numI;i++){
              for(int t=0;t<(le.numT-tao);t++){



    compras2etapa[i][t][iterocur]=occ.compras2[i][t];

              }
              }

    for(int i=0;i<le.numI;i++){
              for(int t=0;t<tao;t++){
                        invtis2[i][t][iterocur]=occ.invtis[i][t];

              backorderis[i][t][iterocur]=occ.backorderis[i][t];
//System.out.println("el BackOrder es
"+backorderis[i][t][iterocur]);
              }
    }

    for(int r=0;r<le.numR;r++){
              for(int t=0;t<le.numT-tao;t++){


    penrecursis1[r][t][iterocur]=occ.penrecursis1[r][t];
              }
    }

    for(int r=0;r<le.numR;r++){
              for(int t=0;t<le.numT-tao;t++){


    penrecursis2[r][t][iterocur]=occ.penrecursis2[r][t];

              }
    }
    //Calculando las ventas reales del sistema
    for(int t=0;t<tao;t++){
              ventis1[t][iterocur]=occ.ventas[t];
    }


    for(int t=0;t<le.numT-tao;t++){

              falta[t][iterocur]=occ.falta[t];
```

```
    }

    for(int t=0;t<le.numT-tao;t++){

              stock[t][iterocur]=occ.stock[t];

    }

    for(int k=0;k<le.numK;k++){
    for(int t=0;t<(le.numT-tao);t++){
              costrokes2[k][t][iterocur]=occ.costrokes2[k][t];

              }
    }

    for(int k=0;k<le.numK;k++){
    for(int t=0;t<(le.numT-tao);t++){
              strokes2etapa[k][t][iterocur]=occ.Stroke2[k][t];

              }
    }

}//Fin Result2

public void ResGeneral(LeerXML le, int tao){

//Relleno faltantes total
              double acumm=0;
    //double [] acumm2=new double[le.numO];
              for(int o=0;o<le.numO;o++){
                        acumm=acumm + (ft1[o]+ft2[o]);

              }
              faltanteTot=(acumm/le.numO);

    //Cálculo la demanda total
              for(int o=0;o<le.numO;o++){
                        for(int i=0;i<le.numI;i++){
                                  for(int t=0;t<le.numT;t++){


    acumm=acumm+le.dEMito[i][t][o];
                                  }
                        }
              }
              NV=1-faltanteTot/(acumm/le.numO);

    //RElleno del valor residual

              for(int t=0;t<le.numT;t++){
    //System.out.println(" es  "+le.numT);
                        if(t>=tao){

                                  double acum=0;
                                  for(int o=0;o<le.numO;o++){

                                            acum=acum +
(VResidual[t-tao][o]);
                                  }
                                  VRGEN[t]=(acum/le.numO);
```

```
                    }
                }

    //Rellenado de la matriz de ventas


                for(int t=0;t<le.numT;t++){
    //System.out.println(" es  "+le.numT);
                    if(t>=tao){

                            double acum=0;
                            for(int o=0;o<le.numO;o++){

                acum=acum + (ventis2[t-tao][o]);
    //Aquí van las ventas reales calculadas de la simulacion

                    }
                    VentasGen[t]=(acum/le.numO);

                }else{
                    double acum=0;
                    for(int o=0;o<le.numO;o++){


    //System.out.println(produceM1[i][t][0]);
                            acum=acum +ventis1[t][o];

                    }
                    VentasGen[t]=(acum/le.numO);


                    }
    //System.out.println(VentasGen[t]);
    }

    //Rellenado de la matriz de la produccion
    for(int i=0;i<le.numI;i++){
                for(int t=0;t<le.numT;t++){
                    //System.out.println(" es  "+le.numT);
                    if(t>=tao){

                    double acum=0;
                    for(int o=0;o<le.numO;o++){

                    acum=acum + (produceM2[i][t-tao][o]);

                    }
                    ProdGen[i][t]=(acum/le.numO);

                }else{
                    //System.out.println(produceM1[i][t][0]);
                    ProdGen[i][t]=produceM1[i][t][0];

                    }
                }
}

    //Rellenado de la matriz de compras
    for(int i=0;i<le.numI;i++){

    for(int t=0;t<le.numT;t++){
```

```
                //System.out.println(" es  "+le.numT);
                if(t>=tao){

                        double acum=0;
                        for(int o=0;o<le.numO;o++){

                                acum=acum + (compras2etapa[i][t-
tao][o]);

                                }
                                ComprasGen[i][t]=(acum/le.numO);

                        }else{

    ComprasGen[i][t]=compras1etapa[i][t][0];
                        }
                }
    }

    //Rellenado de la matriz General de Strokes
    for(int k=0;k<le.numK;k++){
                        for(int t=0;t<le.numT;t++){

    //System.out.println(" es  "+le.numT);
                if(t>=tao){

                        double acum=0;
                        for(int o=0;o<le.numO;o++){

                        acum=acum + (strokes2etapa[k][t-tao][o]);

                        }
                        StrokeGen [k][t]=(acum/le.numO);

                }else{
                        StrokeGen [k][t]=strokes1etapa[k][t];

                }
                }
    }

    //Rellenado de la matriz General de costes de Strokes
    for(int k=0;k<le.numK;k++){

                for(int t=0;t<le.numT;t++){
                        //System.out.println(" es  "+le.numT);
                        if(t>=tao){

                        double acum=0;
                        for(int o=0;o<le.numO;o++){

                        acum=acum + (costrokes2[k][t-tao][o]);

                        }
                        CstrokeGen [k][t]=(acum/le.numO);

                        }else{

                        CstrokeGen [k][t]=costrokes[k][t];

                        }
```

```
            }
    }

    //Rellenado de la matriz General de costes de Penalidad por
subuso de los recursos
    for(int r=0;r<le.numR;r++){
            for(int t=0;t<le.numT;t++){
                    if(t>=tao){
                    double acum=0;
                    for(int o=0;o<le.numO;o++){

                    acum=acum + penrecursis1[r][t-tao][o];


                    }
                    CRPEN1Gen[r][t]=acum/le.numO;
                    }else{

                    CRPEN1Gen[r][t]=0;          //Porque se
fabrica exactamente la capacidad con los stroke
                    }
            }
    }

//Rellenado de la matriz General de costes de Penalidad por
sobreuso de los recursos
    for(int r=0;r<le.numR;r++){
            for(int t=0;t<le.numT;t++){
                    if(t>=tao){
                    double acum=0;
            for(int o=0;o<le.numO;o++){
                    acum=acum + penrecursis2[r][t-tao][o];


                    }
                    CRPEN2Gen [r][t]=acum/le.numO;
                    }else{

                    CRPEN2Gen[r][t]=0;          //Porque se
fabrica exactamente la capacidad con los stroke
                    }
            }
    }

    //Rellenado y cálculo de los costes de compra de materiales
    for(int i=0;i<le.numI;i++){
            for(int t=0;t<le.numT;t++){
            CostCompraGen[i][t]=ComprasGen[i][t]*le.pCit[i][t];
            }
    }

    //Relleno de matriz de costes de inventario

    double [][] valor=new double [tao][le.numO];
    for(int o=0;o<le.numO;o++){

            for(int t=0;t<tao;t++){
                    for(int i=0;i<le.numI;i++){
```

```
valor[t][o]=valor[t][o]+le.cHit[i][t]*invtis2[i][t][o];

                    }

          }
}


for(int t=0;t<le.numT;t++){
          //System.out.println("jejejejejjee "+tao);
          if(t>=tao){
                    double acum=0;
                    for(int o=0;o<le.numO;o++){

                              acum=acum+stock[t-tao][o];

                    }

                    CInventarioGen[t]=acum/le.numO;

          }else{
                    double acum=0;
                    for(int o=0;o<le.numO;o++){

                              acum=acum +valor[t][o];

                    }

                    CInventarioGen[t]=acum/le.numO;
          }
}

          //Rellenando los costes de backorder totales

          for(int t=0;t<le.numT;t++){

          if(t>=tao){
                    double acum=0;

                    for(int o=0;o<le.numO;o++){

                              acum=acum+falta[t-tao][o];
                                                        }

                    CBackOrderGen[t]=acum/le.numO;

//System.out.println("EL coste promedio es
"+CBackOrderGen[t]);
          }else{
                    double acum=0;
                    for(int o=0;o<le.numO;o++){

                              for(int i=0;i<le.numI;i++){



acum=acum+le.cBit[i][t]*backorderis[i][t][o];
                              }
```

```
                    }

                    CBackOrderGen[t]=acum/le.numO;
            }

   }//Fin for Backorders

   //Cálculo del coste total por perido
   for(int t=0;t<le.numT;t++){
            double Cstroke=0;
            double Crecursos=0;
            double Ccompras=0;
            double CInBak=0;

            for(int i=0;i<le.numI;i++){
                    Ccompras=Ccompras+CostCompraGen[i][t];
            }
            //Costes de Strokes
            for(int k=0;k<le.numK;k++){
                    Cstroke=Cstroke+CstrokeGen[k][t];

            }
            //Coste de recursos
            for(int r=0;r<le.numR;r++){
Crecursos=Crecursos+CRPEN1Gen[r][t]+CRPEN2Gen[r][t]+CosRecursos
Gen[r][t];
            }

//Costes de inventario y Backorder
            CInBak=CInventarioGen[t]+CBackOrderGen[t];
            CostTotGen[t]= CInBak+Cstroke+Crecursos+Ccompras;

//System.out.println("SStroke "+ CostTotGen[t]);
   }

   //Cálculo del Beneficio
   for(int t=0;t<le.numT;t++){
            benesperado[t]=VentasGen[t]+VRGEN[t]-CostTotGen[t];
   }


   public void Escribir(LeerXML le, String nombre){

        try{
            // Aquí se comienza a escribir el constructor de la
información
            WritableWorkbook libro =
Workbook.createWorkbook(new File(nombre));
            // Se crea un nuevo libro
            //Se entra en una iteración que depende del número
de escenarios

//-------------------------------------------------------------
-
                WritableSheet hoja ;
                hoja = libro.createSheet("P & G", 0);
                Label label;
//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(4,2,"Ventas totales" ); // Se crea el Label
```

```
                hoja.addCell(label);// Se escribe el Label en
la celda
label = new Label(5,2,"Valor Residual del inv" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
label = new Label(6,2,"Costes Totales" ); // Se crea el Label
hoja.addCell(label);// Se escribe el Label en la celda
label = new Label(7,2,"Beneficio Esperado" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
// Costes de Strokes
for(int t=0; t<le.numT; t++){
    Number number =new Number(3,3+t, (t+1));
     hoja.addCell(number);
}
//Escribir Los Costes de Stroke por periodo

for(int t=0; t<le.numT; t++){
   Number number =new Number(4,3+t, VentasGen[t]);
   hoja.addCell(number);
   number =new Number(5,3+t, VRGEN[t]);
   hoja.addCell(number);
   number =new Number(6,3+t, CostTotGen[t]);
   hoja.addCell(number);
   number =new Number(7,3+t, benesperado[t]);
   hoja.addCell(number);

}
label = new Label(0,0,"El nivel de servicio es: " ); // Se crea
el Label
hoja.addCell(label);// Se escribe el Label en la celda
Number number2 =new Number(2,0, NV);
hoja.addCell(number2);


//---------------------------------------------------------------
---------
   hoja = libro.createSheet("Strokes & SKUs", 1);
//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(0,0,"Costes Promedio de Strokes por Peridos"
); // Se crea el Label
hoja.addCell(label);// Se escribe el Label en la celda
// Costes de Strokes
for(int t=0; t<le.numT; t++){
   Number number =new Number(t+4,2, (t+1));
hoja.addCell(number);
}
//Escribir Los Costes de Stroke por periodo
for(int k=0; k<le.numK; k++){
   label = new Label(3,3+k, le.getStroke[k]);
hoja.addCell(label);
   for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+k, CstrokeGen[k][t]);
hoja.addCell(number);
   }
}

label = new Label(0,5+le.numK,"Strokes Promedios Ejecutados por
periodos" ); // Se crea el Label
hoja.addCell(label);// Se escribe el Label en la celda
```

```
for(int t=0; t<le.numT; t++){
    Number number =new Number(t+4,2+4+le.numK, (t+1));
                        hoja.addCell(number);
                    }
//Escribir los Stroke por periodo
                     for(int k=0; k<le.numK; k++){

label = new Label(3,3+k+4+le.numK, le.getStroke[k]);
                        hoja.addCell(label);

for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+k+4+le.numK,
(StrokeGen[k][t]));
//System.out.println("El valor es "+le.cOkt[k][t]+" es
"+CstrokeGen[k][t]/001);
                            hoja.addCell(number);
                      }
                    }

label = new Label(0,9+2*le.numK,"Generación Promedio de
Productos por periodos" ); // Se crea el Label
hoja.addCell(label);// Se escribe el Label en la celda

 //Escribir los Stroke por periodo
            for(int i=0; i<le.numI; i++){

label = new Label(3,3+i+4+2*le.numK+4, le.getSKU[i]);
                        hoja.addCell(label);

            for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+i+4+2*le.numK+4,
(ProdGen[i][t]));
//System.out.println("El valor es "+le.cOkt[k][t]+" es
"+CstrokeGen[k][t]/001);
                        hoja.addCell(number);
                      }
                    }

                    label = new Label(0,9+2*le.numK + le.numI
+4,"Compras Promedio de Materia Prima por periodos" ); // Se
crea el Label
hoja.addCell(label);// Se escribe el Label en la celda

            //Escribir los Stroke por periodo
            for(int i=0; i<le.numI; i++){

label = new Label(3,3+i+4+2*le.numK+4+le.numI+4, le.getSKU[i]);
                        hoja.addCell(label);

            for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+i+4+2*le.numK+4+le.numI+4,
(ComprasGen[i][t]));
//System.out.println("El valor es "+le.cOkt[k][t]+" es
"+CstrokeGen[k][t]/001);
                            hoja.addCell(number);
                      }
                    }
                    label = new
Label(0,3+4+2*le.numK+4+2*le.numI+6,"Costes de Compras Promedio
por periodos" );
```

```
            hoja.addCell(label);
   //Escribir los costes promedios de compras
            for(int i=0; i<le.numI; i++){

label = new Label(3,3+i+4+2*le.numK+4+2*le.numI+8,
le.getSKU[i]);
                hoja.addCell(label);

                    for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+i+4+2*le.numK+4+2*le.numI+8,
(CostCompraGen[i][t]));
//System.out.println("El valor es "+le.cOkt[k][t]+" es
"+CstrokeGen[k][t]/001);
                                hoja.addCell(number);
                    }
            }

//-------------------------------------------------------------
---
            hoja = libro.createSheet("Coste de los recursos",
2);
//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(0,0,"Costes de Recursos" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
           // Costes de Recursos
             for(int t=0; t<le.numT; t++){
                    Number number =new Number(t+4,2, (t+1));
                 hoja.addCell(number);
             }
           //Escribir Los inventarios totales por periodo
             for(int r=0; r<le.numR; r++){

                    label = new Label(3,3+r,
le.getRecursos[r]);
                hoja.addCell(label);

                    for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+r, CosRecursosGen[r][t]);
                    hoja.addCell(number);
                    }
             }
//-------------------------------------------------------------
---

             hoja = libro.createSheet("Coste Inventario y
Faltantes", 3);
//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(4,2,"Costes de Inventario" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
label = new Label(5,2,"Costes de Faltantes" ); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
// Costes de Strokes
             for(int t=0; t<le.numT; t++){
                    Number number =new Number(3,3+t, (t+1));
                       hoja.addCell(number);
                  }
```

```
                    //Escribir Los Costes de Stroke por periodo

                    for(int t=0; t<le.numT; t++){
Number number =new Number(4,3+t, CInventarioGen[t]);
                     hoja.addCell(number);
                     number =new Number(5,3+t,
CBackOrderGen[t]);
                     hoja.addCell(number);

                    }
//-------------------------------------------------------------
---

                    hoja = libro.createSheet("Penalidad Sobre
uso", 4);
//Se crea una hoja dentro del libro de nombre hojanueva en la
posición 0.
label = new Label(0,0,"Penalidad Sobre Uso"); // Se crea el
Label
hoja.addCell(label);// Se escribe el Label en la celda
      // Costes de Recursos
             for(int t=0; t<le.numT; t++){

                     Number number =new Number(t+4,2, (t+1));
                     hoja.addCell(number);
             }

   //Escribir Los inventarios totales por periodo
             for(int r=0; r<le.numR; r++){

                     label = new Label(3,3+r,
le.getRecursos[r]);
                     hoja.addCell(label);

                     for(int t=0; t<le.numT; t++){
Number number =new Number(t+4,3+r, CRPEN1Gen[r][t]);
                              hoja.addCell(number);
                     }
             }
//-------------------------------------------------

             hoja = libro.createSheet("Penalidad Sub uso", 5);

//Se crea una hoja dentro del libro de nombre ojanueva en la
posición 0.
label = new Label(0,0,"Penalidad Sub Uso" ); // Se rea el Label
hoja.addCell(label);// Se escribe el Label en la elda

   // Costes de Recursos
            for(int t=0; t<le.numT; t++){
                     Number number =new Number(t+4,2, (t+1));
                     hoja.addCell(number);
             }

//Escribir Los inventarios totales por periodo
             for(int r=0; r<le.numR; r++){

                     label = new Label(3,3+r,
le.getRecursos[r]);
                     hoja.addCell(label);
```

```
                    for(int t=0; t<le.numT; t++){

                       Number number =new Number(t+4,3+r,
RPEN2Gen[r][t]);
                    hoja.addCell(number);
                    }
                }
   //-------------------------------------------------------------

//Aquí van los métodos de la hoja de cálculo resumen

//Se escribe el libro
         libro.write();
                libro.close();

                System.out.println("Escribio en Excel");

        }catch(Exception eP){

                    eP.printStackTrace();
        }//Fin Catch
   }}
```

Clase para resolución del segundo modelo matemático (Fase II)

```
package EstocaDET;

import lpsolve.LpSolve;
import EstocaDET.*;

public class SolveM2 {

   public double [][][]x2 ;
   public double [][]PNN;
   public double [][]PNP;
   public double [][][]y2 ;
   public double [][][]b2 ;
   public double FObjetivoE2;
   public double [][] costrokes2;
   public double [][] penrecursis1;
   public double [][] penrecursis2;
   public double [] falta;
   public double [] stock;
   public double [] benesperado2;
   public double [][] produccion2;
   public double [][]stroke2;
   public double [][] compras2;
   public double [] ventas2;
   public double [] VR;
   public double faltTot;


   public double[][] getCompras2() {
            return compras2;
   }

   public SolveM2() {
            super();
   }
```

```
public  void solveM2(LeerXML le, String nombre)throws
Exception{

   LpSolve lp;
   int ret =0;

   boolean sol =false;
   double[] row =new double[1];
   lp=LpSolve.makeLp(0, 0);


lp=LpSolve.readXLI("xli_MathProg", "ModelM2.mod",nombre , "",
0);

   if (lp.getLp() ==0){

   ret =1;// No se ha podido crear el modelo matemático
   }

   if(ret==0){

//El objetivo es maximizar la funcion objetivo

            lp.setVerbose(LpSolve.IMPORTANT);
            System.out.println("Resolviendo el problema");



            lp.setMipGap(true, 0);


   ret = lp.solve(); // Para que LpSolve calcule la Soccion

            String Valor;
            if(ret ==0){
                     Valor = "Optimo";
            }else{
                     Valor = "No va";
            }


   System.out.println("la Salida del modelo es: " + Valor);


            if (lp.solve() ==2){
                     sol = false;
            }

            if (ret == LpSolve.OPTIMAL || ret ==1)
                     ret =0;
            else
                     ret =5;

    }//Fin If

   if(ret==0){ // if 2
            int scalemode =lp.getScaling();
            System.out.println("La escala es: "+ scalemode);
/* La solucion es calculado, ahora obtenemos los resultados*/
```

```
   /*Funcion Objetivo que se desea optimizar */

System.out.println("La funcion objetivo es: " +
lp.getObjective());

   // Valor de las variables

            row = new double[lp.getNcolumns()];
            lp.getVariables(row);

            String [] colName = new String[lp.getNcolumns()];
        for(int i=0; i<colName.length;i++){

             colName[i] = lp.getColName(i+1);
             }

// Escribir los valores de capacidades a renegociar. Si mayor o
menor

            //Tamano del horizonte de planeacion
            int tamano=le.numT-le.tao;

            //Extraer Capacidades Contratadas
            PNN = new double [le.numR][tamano];

            for(int r=0; r<le.numR;r++){

                     for(int t=0; t<tamano; t++){
            //Penalidad por sub uso de la capacidad
                                          PNN [r][t] =
(devuelveValor(colName,row,"FN["+le.getRecursos[r]+","+le.getPe
riodos[t+le.tao]+"]"));

            }
   }

//Extraer Capacidades Contratadas
   PNP = new double [le.numR][tamano];

   for(int r=0; r<le.numR;r++){

    for(int t=0; t<tamano; t++){

PNP [r][t] =
(devuelveValor(colName,row,"FP["+le.getRecursos[r]+","+le.getPe
riodos[t+le.tao]+"]"));

            }
   }

costrokes2 = new double [le.numK][tamano];

for(int k=0; k<le.numK;k++){

   for(int t=0; t<tamano; t++){

costrokes2 [k][t] =
(devuelveValor(colName,row,"SolStroke2["+le.getStroke[k]+","+le
.getPeriodos[t+le.tao]+"]"));
```

```
                }
    }

    stroke2 = new double [le.numK][tamano];

for(int k=0; k<le.numK;k++){

    for(int t=0; t<tamano; t++){

stroke2 [k][t] =
(devuelveValor(colName,row,"z["+le.getStroke[k]+","+le.getPerio
dos[t+le.tao]+"]"));

                }
}

    benesperado2=new double [tamano];
    for(int t=0; t<tamano; t++){

benesperado2 [t] =
(devuelveValor(colName,row,"benef2["+le.getPeriodos[t+le.tao]+"
]"));
    }

    falta=new double [tamano];
       for(int t=0; t<tamano; t++){

falta [t] =
(devuelveValor(colName,row,"falta["+le.getPeriodos[t+le.tao]+"]
"));
                        }

                        stock=new double [tamano];
                        for(int t=0; t<tamano; t++){

stock[t] =
(devuelveValor(colName,row,"stock["+le.getPeriodos[t+le.tao]+"]
"));

                        }

                        penrecursis1 = new double
[le.numR][tamano];
            for(int r=0; r<le.numR;r++){

                         for(int t=0; t<tamano; t++){

penrecursis1 [r][t] =
(devuelveValor(colName,row,"PenTotal1RT["+le.getRecursos[r]+","
+le.getPeriodos[t+le.tao]+"]"));

                         }
            }

            penrecursis2 = new double [le.numR][tamano];

            for(int r=0; r<le.numR;r++){

                         for(int t=0; t<tamano; t++){
```

```
penrecursis2 [r][t] =
(devuelveValor(colName,row,"PenTotal2RT["+le.getRecursos[r]+","
+le.getPeriodos[t+le.tao]+"]"));
            //System.out.println("Penalidad FN Sub Uso
"+penrecursis2 [r][t]);
                        }
            }

            produccion2 = new double [le.numI][tamano];

            for(int i=0; i<le.numI;i++){

                        for(int t=0; t<tamano; t++){

produccion2 [i][t] =
(devuelveValor(colName,row,"produccion["+le.getSKU[i]+","+le.ge
tPeriodos[t+le.tao]+"]"));

                        }
            }

            compras2 = new double [le.numI][tamano];

   for(int i=0; i<le.numI;i++){

                        for(int t=0; t<tamano; t++){

compras2 [i][t] =
(devuelveValor(colName,row,"w["+le.getSKU[i]+","+le.getPeriodos
[t+le.tao]+"]"));

                        }
               }

            ventas2= new double [tamano];


             for(int t=0; t<tamano; t++){

ventas2[t]
(devuelveValor(colName,row,"ventaEsp["+le.getPeriodos[t+le.tao]
+"]"));
    }

   VR= new double [tamano];
   for(int t=0; t<tamano; t++){

VR[t] =
(devuelveValor(colName,row,"vr["+le.getPeriodos[t+le.tao]+"]"))
;
    }

   faltTot = (devuelveValor(colName,row,"fT"));


   FObjetivoE2=lp.getObjective();

//lp.setOutputfile("log.txt");
            // Se extraen los valores pertinentes para el
analisís
```

```
}//Fin If 2

System.out.println("Problema Resuelto");
System.out.println("---------------------------------");
   System.out.println("");

}//Fin del método


static double devuelveValor(String [] nColumns, double []
vColumns, String column){
double valor  = 0;
int contador = 0;
while(nColumns[contador].compareTo(column)!=0){
  contador++;
  if(contador>=vColumns.length){
      break;
   }
}
if(contador>=vColumns.length){
  System.out.println("ERROR En NOMBRE DE VAR: "+column);
}else{
  valor = vColumns[contador];
}
return valor;

}// Fin del método devuelve valor

public double[][][] getX2() {
return x2;
}

public void setX2(double[][][] x2) {
   this.x2 = x2;
   }

   public double[][] getPNN() {
            return PNN;
   }

   public void setPNN(double[][] pnn) {
            PNN = pnn;
   }

   public double[][] getPNP() {
            return PNP;
   }

   public void setPNP(double[][] pnp) {
            PNP = pnp;
   }

   public double[][][] getY2() {
            return y2;
   }

   public void setY2(double[][][] y2) {
            this.y2 = y2;
   }

   public double[][][] getB2() {
```

```
        return b2;
}

public void setB2(double[][][] b2) {
        this.b2 = b2;
}

public double getFObjetivoE2() {
        return FObjetivoE2;
}

public void setFObjetivoE2(double objetivoE2) {
        FObjetivoE2 = objetivoE2;
}

public double[][] getCostrokes2() {
        return costrokes2;
}

public void setCostrokes2(double[][] costrokes2) {
        this.costrokes2 = costrokes2;
}

public double[][] getPenrecursis1() {
        return penrecursis1;
}

public void setPenrecursis1(double[][] penrecursis1) {
        this.penrecursis1 = penrecursis1;
}

public double[][] getPenrecursis2() {
        return penrecursis2;
}

public void setPenrecursis2(double[][] penrecursis2) {
        this.penrecursis2 = penrecursis2;
}

public double[] getFalta() {
        return falta;
}

public void setFalta(double[] falta) {
        this.falta = falta;
}

public double[] getStock() {
        return stock;
}

public void setStock(double[] stock) {
        this.stock = stock;
}

public double[] getBenesperado2() {
        return benesperado2;
}

public double[][] getProduccion2() {
        return produccion2;
```

```
    }

    public void setBenesperado2(double[] benesperado2) {
            this.benesperado2 = benesperado2;
    }

    public double[][] getStroke2() {
            return stroke2;
    }

    public double[] getVentas2() {
            return ventas2;
    }

    public double[] getVR() {
            return VR;
    }

    public double getFaltTot() {
            return faltTot;
    }
}
```

LECTOR XML

```
//
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v2.1-b02-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source
schema.
// Generated on: 2010.04.15 at 11:52:41 PM CEST
//

package LectorXML;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;
/**
 * <p>Java class for ActivaCompra complex type.
 *
 * <p>The following schema fragment specifies the expected content contained
within this class.
 *
 * <pre>
 * &lt;complexType name="ActivaCompra">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
```

```
 *       &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="ACit"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "ActivaCompra")
public class ActivaCompra {

    @XmlAttribute
    protected String i;
    @XmlAttribute
    protected String t;
    @XmlAttribute(name = "ACit")
    protected Double aCit;

    /**
     * Gets the value of the i property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getI() {
        return i;
    }

    /**
     * Sets the value of the i property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setI(String value) {
        this.i = value;
    }

    /**
     * Gets the value of the t property.
     *
     * @return
```

```
     *     possible object is
     *     {@link String }
     *
     */
    public String getT() {
        return t;
    }

    /**
     * Sets the value of the t property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setT(String value) {
        this.t = value;
    }

    /**
     * Gets the value of the aCit property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getACit() {
        return aCit;
    }

    /**
     * Sets the value of the aCit property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setACit(Double value) {
        this.aCit = value;
    }
}
//
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v2.1-b02-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
```

```
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2010.04.15 at 11:52:41 PM CEST
//

package LectorXML;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>Java class for BOM complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within this class.
 *
 * <pre>
 * &lt;complexType name="BOM">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i" type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="k" type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="Mik" type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "BOM")
public class BOM {

    @XmlAttribute
    protected String i;
    @XmlAttribute
    protected String k;
    @XmlAttribute(name = "Mik")
    protected Double mik;
    /**
     * Gets the value of the i property.
     *
     * @return
     *     possible object is
     *     {@link String }
```

```
 *
 */
public String getI() {
    return i;
}
/**
 * Sets the value of the i property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setI(String value) {
    this.i = value;
}
/**
 * Gets the value of the k property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getK() {
    return k;
}
/**
 * Sets the value of the k property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setK(String value) {
    this.k = value;
}
/**
 * Gets the value of the mik property.
 *
 * @return
 *     possible object is
 *     {@link Double }
 *
 */
public Double getMik() {
    return mik;
}
/**
```

```
    * Sets the value of the mik property.
    *
    * @param value
    *     allowed object is
    *     {@link Double }
    *
    */
   public void setMik(Double value) {
      this.mik = value;
   }
}
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v2.1-b02-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jax
b</a>
// Any modifications to this file will be lost upon
recompilation of the source schema.
    // Generated on: 2010.04.15 at 11:52:41 PM CEST
    //


    package LectorXML;

    import javax.xml.bind.annotation.XmlAccessType;
    import javax.xml.bind.annotation.XmlAccessorType;
    import javax.xml.bind.annotation.XmlAttribute;
    import javax.xml.bind.annotation.XmlType;


    /**
     * <p>Java class for CoefTecno complex type.
     *
     * <p>The following schema fragment specifies the expected
content contained within this class.
     *
     * <pre>
     * &lt;complexType name="CoefTecno">
     *   &lt;complexContent>
     *     &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
*       &lt;attribute name="k"
type="{http://www.w3.org/2001/XMLSchema}string" />
*       &lt;attribute name="r"
type="{http://www.w3.org/2001/XMLSchema}string" />
*       &lt;attribute name="RErk"
type="{http://www.w3.org/2001/XMLSchema}double" />
     *     &lt;/restriction>
     *   &lt;/complexContent>
     * &lt;/complexType>
     * </pre>
     *
     *
     */
    @XmlAccessorType(XmlAccessType.FIELD)
    @XmlType(name = "CoefTecno")
    public class CoefTecno {

        @XmlAttribute
```

```
protected String k;
@XmlAttribute
protected String r;
@XmlAttribute(name = "RErk")
protected Double rErk;

/**
 * Gets the value of the k property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getK() {
    return k;
}

/**
 * Sets the value of the k property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setK(String value) {
    this.k = value;
}

/**
 * Gets the value of the r property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getR() {
    return r;
}

/**
 * Sets the value of the r property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setR(String value) {
    this.r = value;
}

/**
 * Gets the value of the rErk property.
 *
 * @return
 *     possible object is
 *     {@link Double }
 *
```

```
         */
        public Double getRErk() {
            return rErk;
        }

        /**
         * Sets the value of the rErk property.
         *
         * @param value
         *     allowed object is
         *     {@link Double }
         *
         */
        public void setRErk(Double value) {
            this.rErk = value;
        }

    }
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v2.1-b02-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jax
b</a>
// Any modifications to this file will be lost upon
recompilation of the source schema.
// Generated on: 2010.04.15 at 11:52:41 PM CEST
//


package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for CosCapacidad complex type.
 *
 * <p>The following schema fragment specifies the expected
content contained within this class.
 *
 * <pre>
 * &lt;complexType name="CosCapacidad">
 *   &lt;complexContent>
 *     &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="r"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="CCAPrt"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
```

```java
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "CosCapacidad")
public class CosCapacidad {

    @XmlAttribute
    protected String r;
    @XmlAttribute
    protected String t;
    @XmlAttribute(name = "CCAPrt")
    protected Double ccaPrt;

    /**
     * Gets the value of the r property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getR() {
        return r;
    }

    /**
     * Sets the value of the r property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setR(String value) {
        this.r = value;
    }

    /**
     * Gets the value of the t property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getT() {
        return t;
    }

    /**
     * Sets the value of the t property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setT(String value) {
        this.t = value;
    }

    /**
     * Gets the value of the ccaPrt property.
```

```
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getCCAPrt() {
        return ccaPrt;
    }

    /**
     * Sets the value of the ccaPrt property.
     *
* @param value
*     allowed object is
*     {@link Double }
*
*/
public void setCCAPrt(Double value) {
this.ccaPrt = value;
    }

}
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v2.1-b02-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jax
b</a>
// Any modifications to this file will be lost upon
recompilation of the source schema.
// Generated on: 2010.04.15 at 11:52:41 PM CEST
//


package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for CosteBackOrder complex type.
 *
 * <p>The following schema fragment specifies the expected
content contained within this class.
 *
 * <pre>
 * &lt;complexType name="CosteBackOrder">
 *   &lt;complexContent>
 *     &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="CBit"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
```

```
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "CosteBackOrder")
public class CosteBackOrder {

    @XmlAttribute
    protected String i;
    @XmlAttribute
    protected String t;
    @XmlAttribute(name = "CBit")
    protected Double cBit;

    /**
     * Gets the value of the i property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getI() {
        return i;
    }

    /**
     * Sets the value of the i property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setI(String value) {
        this.i = value;
    }

    /**
     * Gets the value of the t property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getT() {
        return t;
    }

    /**
     * Sets the value of the t property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setT(String value) {
```

```
        this.t = value;
    }

    /**
     * Gets the value of the cBit property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getCBit() {
        return cBit;
    }

    /**
     * Sets the value of the cBit property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setCBit(Double value) {
        this.cBit = value;
                    }

}
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v2.1-b02-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jax
b</a>
// Any modifications to this file will be lost upon
recompilation of the source schema.
// Generated on: 2010.04.15 at 11:52:41 PM CEST
//


package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for CosteInventory complex type.
 *
 * <p>The following schema fragment specifies the expected
content contained within this class.
 *
 * <pre>
 * &lt;complexType name="CosteInventory">
 *   &lt;complexContent>
 *     &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
```

```
 *       &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="CH"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "CosteInventory")
public class CosteInventory {

    @XmlAttribute
    protected String i;
    @XmlAttribute
    protected String t;
    @XmlAttribute(name = "CH")
    protected Double ch;

    /**
     * Gets the value of the i property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getI() {
        return i;
    }

    /**
     * Sets the value of the i property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setI(String value) {
        this.i = value;
    }

    /**
     * Gets the value of the t property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getT() {
        return t;
    }

    /**
     * Sets the value of the t property.
     *
```

```
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setT(String value) {
        this.t = value;
    }

    /**
     * Gets the value of the ch property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getCH() {
        return ch;
    }

    /**
     * Sets the value of the ch property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setCH(Double value) {
        this.ch = value;
    }

}
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v2.1-b02-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jax
b</a>
// Any modifications to this file will be lost upon
recompilation of the source schema.
// Generated on: 2010.04.15 at 11:52:41 PM CEST
//


package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for CosteStroke complex type.
 *
 * <p>The following schema fragment specifies the expected
content contained within this class.
 *
 * <pre>
 * &lt;complexType name="CosteStroke">
```

```
 *    &lt;complexContent>
 *      &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
 *        &lt;attribute name="k"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *        &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *        &lt;attribute name="COkt"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *      &lt;/restriction>
 *    &lt;/complexContent>
 *  &lt;/complexType>
 *  </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "CosteStroke")
public class CosteStroke {

    @XmlAttribute
    protected String k;
    @XmlAttribute
    protected String t;
    @XmlAttribute(name = "COkt")
    protected Double cOkt;

    /**
     * Gets the value of the k property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getK() {
        return k;
    }

    /**
     * Sets the value of the k property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setK(String value) {
        this.k = value;
    }

    /**
     * Gets the value of the t property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getT() {
        return t;
```

```
    }

    /**
     * Sets the value of the t property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setT(String value) {
        this.t = value;
    }

    /**
     * Gets the value of the cOkt property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getCOkt() {
        return cOkt;
    }

    /**
     * Sets the value of the cOkt property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setCOkt(Double value) {
        this.cOkt = value;
    }

}
```

// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, v2.1-b02-fcs
// See http://java.sun.com/xml/jaxb
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2010.04.15 at 11:52:41 PM CEST
//

```
package LectorXML;
import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;
```

```
/**
 * <p>Java class for anonymous complex type.
 *
 * <p>The following schema fragment specifies the expected content contained
within this class.
 *
 * <pre>
 * &lt;complexType>
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;sequence>
 *         &lt;element name="DemEsS"
type="{http://www.example.org/Entradas}DemandaEs"
maxOccurs="unbounded"/>
 *         &lt;element name="DemRealS"
type="{http://www.example.org/Entradas}DemandaRe"
maxOccurs="unbounded"/>
 *         &lt;element name="Precios"
type="{http://www.example.org/Entradas}PrecioProd"
maxOccurs="unbounded"/>
 *         &lt;element name="ValorResidualI"
type="{http://www.example.org/Entradas}ResidualInv"
maxOccurs="unbounded"/>
 *         &lt;element name="CosBackorders"
type="{http://www.example.org/Entradas}CosteBackOrder"
maxOccurs="unbounded"/>
 *         &lt;element name="PrecioCompras"
type="{http://www.example.org/Entradas}PrecioCompra"
maxOccurs="unbounded"/>
 *         &lt;element name="CosStrokes"
type="{http://www.example.org/Entradas}CosteStroke"
maxOccurs="unbounded"/>
 *         &lt;element name="CostesCapacidad"
type="{http://www.example.org/Entradas}CosCapacidad"
maxOccurs="unbounded"/>
 *         &lt;element name="CosteAlmacena"
type="{http://www.example.org/Entradas}CosteInventory"
maxOccurs="unbounded"/>
 *         &lt;element name="BOMs"
type="{http://www.example.org/Entradas}BOM" maxOccurs="unbounded"/>
 *         &lt;element name="rBOMs"
type="{http://www.example.org/Entradas}rBOM" maxOccurs="unbounded"/>
 *         &lt;element name="CoeRecurs"
type="{http://www.example.org/Entradas}CoefTecno"
maxOccurs="unbounded"/>
 *         &lt;element name="Compras"
type="{http://www.example.org/Entradas}ActivaCompra"
maxOccurs="unbounded"/>
```

```
 *         &lt;element name="LimiteRecursos"
type="{http://www.example.org/Entradas}UpperBounCapacidad"
maxOccurs="unbounded"/>
 *         &lt;element name="PenalidadesMas"
type="{http://www.example.org/Entradas}PenalidadMAS"
maxOccurs="unbounded"/>
 *         &lt;element name="PenalidadesMenos"
type="{http://www.example.org/Entradas}PenalidadMENOS"
maxOccurs="unbounded"/>
 *         &lt;element name="Probabilidades"
type="{http://www.example.org/Entradas}ProbabilidadEsc"
maxOccurs="unbounded"/>
 *         &lt;element name="ConjuntoSKU"
type="{http://www.w3.org/2001/XMLSchema}string"
maxOccurs="unbounded"/>
 *         &lt;element name="ConjuntoEscenarios"
type="{http://www.w3.org/2001/XMLSchema}string"
maxOccurs="unbounded"/>
 *         &lt;element name="CojuntoStrokes"
type="{http://www.w3.org/2001/XMLSchema}string"
maxOccurs="unbounded"/>
 *         &lt;element name="ConjuntoRecursos"
type="{http://www.w3.org/2001/XMLSchema}string"
maxOccurs="unbounded"/>
 *         &lt;element name="ConjuntoPeriodos"
type="{http://www.w3.org/2001/XMLSchema}string"
maxOccurs="unbounded"/>
 *         &lt;element name="ConjuntoEventos"
type="{http://www.w3.org/2001/XMLSchema}string"
maxOccurs="unbounded"/>
 *         &lt;element name="Leadtimes"
type="{http://www.example.org/Entradas}leadtime"
maxOccurs="unbounded"/>
 *         &lt;element name="InvInicial"
type="{http://www.example.org/Entradas}InventarioIncial"
maxOccurs="unbounded"/>
 *         &lt;element name="DemandasBass"
type="{http://www.example.org/Entradas}DemBass"
maxOccurs="unbounded"/>
 *       &lt;/sequence>
 *       &lt;attribute name="tao"
type="{http://www.w3.org/2001/XMLSchema}int" />
 *       &lt;attribute name="numI"
type="{http://www.w3.org/2001/XMLSchema}int" />
 *       &lt;attribute name="numT"
type="{http://www.w3.org/2001/XMLSchema}int" />
 *       &lt;attribute name="numR"
type="{http://www.w3.org/2001/XMLSchema}int" />
 *       &lt;attribute name="numK"
type="{http://www.w3.org/2001/XMLSchema}int" />
```

```
 *       &lt;attribute name="numE"
type="{http://www.w3.org/2001/XMLSchema}int" />
 *       &lt;attribute name="Tipo"
type="{http://www.w3.org/2001/XMLSchema}int" />
 *       &lt;attribute name="numO"
type="{http://www.w3.org/2001/XMLSchema}int" />
 *       &lt;attribute name="iniciaDemanda"
type="{http://www.w3.org/2001/XMLSchema}int" />
 *  .   &lt;/restriction>
 *  … &lt;/complexContent>
 *….. &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "demEsS",
    "demRealS",
    "precios",
    "valorResidualI",
    "cosBackorders",
    "precioCompras",
    "cosStrokes",
    "costesCapacidad",
    "costeAlmacena",
    "boMs",
    "rboMs",
    "coeRecurs",
    "compras",
    "limiteRecursos",
    "penalidadesMas",
    "penalidadesMenos",
    "probabilidades",
    "conjuntoSKU",
    "conjuntoEscenarios",
    "cojuntoStrokes",
    "conjuntoRecursos",
    "conjuntoPeriodos",
    "conjuntoEventos",
    "leadtimes",
    "invInicial",
    "demandasBass"
})
@XmlRootElement(name = "Data")
public class Data {

    @XmlElement(name = "DemEsS", required = true)
    protected List<DemandaEs> demEsS;
    @XmlElement(name = "DemRealS", required = true)
```

```
protected List<DemandaRe> demRealS;
@XmlElement(name = "Precios", required = true)
protected List<PrecioProd> precios;
@XmlElement(name = "ValorResidualI", required = true)
protected List<ResidualInv> valorResidualI;
@XmlElement(name = "CosBackorders", required = true)
protected List<CosteBackOrder> cosBackorders;
@XmlElement(name = "PrecioCompras", required = true)
protected List<PrecioCompra> precioCompras;
@XmlElement(name = "CosStrokes", required = true)
protected List<CosteStroke> cosStrokes;
@XmlElement(name = "CostesCapacidad", required = true)
protected List<CosCapacidad> costesCapacidad;
@XmlElement(name = "CosteAlmacena", required = true)
protected List<CosteInventory> costeAlmacena;
@XmlElement(name = "BOMs", required = true)
protected List<BOM> boMs;
@XmlElement(name = "rBOMs", required = true)
protected List<RBOM> rboMs;
@XmlElement(name = "CoeRecurs", required = true)
protected List<CoefTecno> coeRecurs;
@XmlElement(name = "Compras", required = true)
protected List<ActivaCompra> compras;
@XmlElement(name = "LimiteRecursos", required = true)
protected List<UpperBounCapacidad> limiteRecursos;
@XmlElement(name = "PenalidadesMas", required = true)
protected List<PenalidadMAS> penalidadesMas;
@XmlElement(name = "PenalidadesMenos", required = true)
protected List<PenalidadMENOS> penalidadesMenos;
@XmlElement(name = "Probabilidades", required = true)
protected List<ProbabilidadEsc> probabilidades;
@XmlElement(name = "ConjuntoSKU", required = true)
protected List<String> conjuntoSKU;
@XmlElement(name = "ConjuntoEscenarios", required = true)
protected List<String> conjuntoEscenarios;
@XmlElement(name = "CojuntoStrokes", required = true)
protected List<String> cojuntoStrokes;
@XmlElement(name = "ConjuntoRecursos", required = true)
protected List<String> conjuntoRecursos;
@XmlElement(name = "ConjuntoPeriodos", required = true)
protected List<String> conjuntoPeriodos;
@XmlElement(name = "ConjuntoEventos", required = true)
protected List<String> conjuntoEventos;
@XmlElement(name = "Leadtimes", required = true)
protected List<Leadtime> leadtimes;
@XmlElement(name = "InvInicial", required = true)
protected List<InventarioIncial> invInicial;
@XmlElement(name = "DemandasBass", required = true)
protected List<DemBass> demandasBass;
@XmlAttribute
```

```
    protected Integer tao;
    @XmlAttribute
    protected Integer numI;
    @XmlAttribute
    protected Integer numT;
    @XmlAttribute
    protected Integer numR;
    @XmlAttribute
    protected Integer numK;
    @XmlAttribute
    protected Integer numE;
    @XmlAttribute(name = "Tipo")
    protected Integer tipo;
    @XmlAttribute
    protected Integer numO;
    @XmlAttribute
    protected Integer iniciaDemanda;

    /**
     * Gets the value of the demEsS property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the demEsS
property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getDemEsS().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link DemandaEs }
     *
     *
     */
    public List<DemandaEs> getDemEsS() {
        if (demEsS == null) {
            demEsS = new ArrayList<DemandaEs>();
        }
        return this.demEsS;
    }
    /**
     * Gets the value of the demRealS property.
     *
```

```
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the demRealS
property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getDemRealS().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link DemandaRe }
     *
     *
     */
    public List<DemandaRe> getDemRealS() {
        if (demRealS == null) {
            demRealS = new ArrayList<DemandaRe>();
        }
        return this.demRealS;
    }
    /**
     * Gets the value of the precios property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the precios
property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getPrecios().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link PrecioProd }
     *
     *
     */
    public List<PrecioProd> getPrecios() {
```

```
        if (precios == null) {
            precios = new ArrayList<PrecioProd>();
        }
        return this.precios;
    }
    /**
     * Gets the value of the valorResidualI property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
valorResidualI property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getValorResidualI().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link ResidualInv }
     *
     *
     */
    public List<ResidualInv> getValorResidualI() {
        if (valorResidualI == null) {
            valorResidualI = new ArrayList<ResidualInv>();
        }
        return this.valorResidualI;
    }
    /**
     * Gets the value of the cosBackorders property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
cosBackorders property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getCosBackorders().add(newItem);
     * </pre>
     *
```

```
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link CosteBackOrder }
     *
     *
     */
    public List<CosteBackOrder> getCosBackorders() {
        if (cosBackorders == null) {
            cosBackorders = new ArrayList<CosteBackOrder>();
        }
        return this.cosBackorders;
    }
    /**
     * Gets the value of the precioCompras property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
precioCompras property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getPrecioCompras().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link PrecioCompra }
     *
     *
     */
    public List<PrecioCompra> getPrecioCompras() {
        if (precioCompras == null) {
            precioCompras = new ArrayList<PrecioCompra>();
        }
        return this.precioCompras;
    }
    /**
     * Gets the value of the cosStrokes property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
```

```
     * This is why there is not a <CODE>set</CODE> method for the
cosStrokes property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getCosStrokes().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link CosteStroke }
     *
     *
     */
    public List<CosteStroke> getCosStrokes() {
        if (cosStrokes == null) {
            cosStrokes = new ArrayList<CosteStroke>();
        }
        return this.cosStrokes;
    }
    /**
     * Gets the value of the costesCapacidad property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
costesCapacidad property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getCostesCapacidad().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link CosCapacidad }
     *
     *
     */
    public List<CosCapacidad> getCostesCapacidad() {
        if (costesCapacidad == null) {
            costesCapacidad = new ArrayList<CosCapacidad>();
        }
        return this.costesCapacidad;
```

```
    }
    /**
     * Gets the value of the costeAlmacena property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
costeAlmacena property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getCosteAlmacena().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link CosteInventory }
     *
     *
     */
    public List<CosteInventory> getCosteAlmacena() {
        if (costeAlmacena == null) {
            costeAlmacena = new ArrayList<CosteInventory>();
        }
        return this.costeAlmacena;
    }
    /**
     * Gets the value of the boMs property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the boMs
property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getBOMs().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link BOM }
```

```
     *
     *
     */
    public List<BOM> getBOMs() {
        if (boMs == null) {
            boMs = new ArrayList<BOM>();
        }
        return this.boMs;
    }
    /**
     * Gets the value of the rboMs property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the rboMs
property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getRBOMs().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link RBOM }
     *
     *
     */
    public List<RBOM> getRBOMs() {
        if (rboMs == null) {
            rboMs = new ArrayList<RBOM>();
        }
        return this.rboMs;
    }
    /**
     * Gets the value of the coeRecurs property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
coeRecurs property.
     *
     * <p>
     * For example, to add a new item, do as follows:
```

```
     * <pre>
     *    getCoeRecurs().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link CoefTecno }
     *
     *
     */
    public List<CoefTecno> getCoeRecurs() {
        if (coeRecurs == null) {
            coeRecurs = new ArrayList<CoefTecno>();
        }
        return this.coeRecurs;
    }
    /**
     * Gets the value of the compras property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the compras
property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getCompras().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link ActivaCompra }
     *
     *
     */
    public List<ActivaCompra> getCompras() {
        if (compras == null) {
            compras = new ArrayList<ActivaCompra>();
        }
        return this.compras;
    }
    /**
     * Gets the value of the limiteRecursos property.
     *
     * <p>
```

```
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
limiteRecursos property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getLimiteRecursos().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link UpperBounCapacidad }
     *
     *
     */
    public List<UpperBounCapacidad> getLimiteRecursos() {
        if (limiteRecursos == null) {
            limiteRecursos = new ArrayList<UpperBounCapacidad>();
        }
        return this.limiteRecursos;
    }
    /**
     * Gets the value of the penalidadesMas property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
penalidadesMas property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getPenalidadesMas().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link PenalidadMAS }
     *
     *
     */
    public List<PenalidadMAS> getPenalidadesMas() {
        if (penalidadesMas == null) {
```

```
            penalidadesMas = new ArrayList<PenalidadMAS>();
        }
        return this.penalidadesMas;
    }
    /**
     * Gets the value of the penalidadesMenos property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
penalidadesMenos property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getPenalidadesMenos().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link PenalidadMENOS }
     *
     *
     */
    public List<PenalidadMENOS> getPenalidadesMenos() {
        if (penalidadesMenos == null) {
            penalidadesMenos = new ArrayList<PenalidadMENOS>();
        }
        return this.penalidadesMenos;
    }
    /**
     * Gets the value of the probabilidades property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
probabilidades property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getProbabilidades().add(newItem);
     * </pre>
     *
     *
```

```
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link ProbabilidadEsc }
     *
     *
     */
    public List<ProbabilidadEsc> getProbabilidades() {
        if (probabilidades == null) {
            probabilidades = new ArrayList<ProbabilidadEsc>();
        }
        return this.probabilidades;
    }
    /**
     * Gets the value of the conjuntoSKU property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
conjuntoSKU property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getConjuntoSKU().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link String }
     *
     *
     */
    public List<String> getConjuntoSKU() {
        if (conjuntoSKU == null) {
            conjuntoSKU = new ArrayList<String>();
        }
        return this.conjuntoSKU;
    }
    /**
     * Gets the value of the conjuntoEscenarios property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
conjuntoEscenarios property.
```

```
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getConjuntoEscenarios().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link String }
     *
     *
     */
    public List<String> getConjuntoEscenarios() {
        if (conjuntoEscenarios == null) {
            conjuntoEscenarios = new ArrayList<String>();
        }
        return this.conjuntoEscenarios;
    }
    /**
     * Gets the value of the cojuntoStrokes property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
cojuntoStrokes property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getCojuntoStrokes().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link String }
     *
     *
     */
    public List<String> getCojuntoStrokes() {
        if (cojuntoStrokes == null) {
            cojuntoStrokes = new ArrayList<String>();
        }
        return this.cojuntoStrokes;
    }
    /**
```

```
     * Gets the value of the conjuntoRecursos property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
conjuntoRecursos property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getConjuntoRecursos().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link String }
     *
     *
     */
    public List<String> getConjuntoRecursos() {
        if (conjuntoRecursos == null) {
            conjuntoRecursos = new ArrayList<String>();
        }
        return this.conjuntoRecursos;
    }
    /**
     * Gets the value of the conjuntoPeriodos property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
conjuntoPeriodos property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getConjuntoPeriodos().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link String }
     *
     *
```

```
     */
    public List<String> getConjuntoPeriodos() {
        if (conjuntoPeriodos == null) {
            conjuntoPeriodos = new ArrayList<String>();
        }
        return this.conjuntoPeriodos;
    }
    /**
     * Gets the value of the conjuntoEventos property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
conjuntoEventos property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getConjuntoEventos().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link String }
     *
     *
     */
    public List<String> getConjuntoEventos() {
        if (conjuntoEventos == null) {
            conjuntoEventos = new ArrayList<String>();
        }
        return this.conjuntoEventos;
    }
    /**
     * Gets the value of the leadtimes property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the leadtimes
property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getLeadtimes().add(newItem);
```

```
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link Leadtime }
     *
     *
     */
    public List<Leadtime> getLeadtimes() {
        if (leadtimes == null) {
            leadtimes = new ArrayList<Leadtime>();
        }
        return this.leadtimes;
    }
    /**
     * Gets the value of the invInicial property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the invInicial
property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getInvInicial().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link InventarioIncial }
     *
     *
     */
    public List<InventarioIncial> getInvInicial() {
        if (invInicial == null) {
            invInicial = new ArrayList<InventarioIncial>();
        }
        return this.invInicial;
    }
    /**
     * Gets the value of the demandasBass property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
```

```
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
demandasBass property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getDemandasBass().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link DemBass }
     *
     *
     */
    public List<DemBass> getDemandasBass() {
        if (demandasBass == null) {
            demandasBass = new ArrayList<DemBass>();
        }
        return this.demandasBass;
    }
    /**
     * Gets the value of the tao property.
     *
     * @return
     *     possible object is
     *     {@link Integer }
     *
     */
    public Integer getTao() {
        return tao;
    }
    /**
     * Sets the value of the tao property.
     *
     * @param value
     *     allowed object is
     *     {@link Integer }
     *
     */
    public void setTao(Integer value) {
        this.tao = value;
    }
    /**
     * Gets the value of the numI property.
     *
     * @return
     *     possible object is
```

```
     *     {@link Integer }
     *
     */
    public Integer getNumI() {
        return numI;
    }
    /**
     * Sets the value of the numI property.
     *
     * @param value
     *     allowed object is
     *     {@link Integer }
     *
     */
    public void setNumI(Integer value) {
        this.numI = value;
    }
    /**
     * Gets the value of the numT property.
     *
     * @return
     *     possible object is
     *     {@link Integer }
     *
     */
    public Integer getNumT() {
        return numT;
    }
    /**
     * Sets the value of the numT property.
     *
     * @param value
     *     allowed object is
     *     {@link Integer }
     *
     */
    public void setNumT(Integer value) {
        this.numT = value;
    }
    /**
     * Gets the value of the numR property.
     *
     * @return
     *     possible object is
     *     {@link Integer }
     *
     */
    public Integer getNumR() {
        return numR;
    }
```

```
/**
 * Sets the value of the numR property.
 *
 * @param value
 *     allowed object is
 *     {@link Integer }
 *
 */
public void setNumR(Integer value) {
   this.numR = value;
}
/**
 * Gets the value of the numK property.
 *
 * @return
 *     possible object is
 *     {@link Integer }
 *
 */
public Integer getNumK() {
   return numK;
}
/**
 * Sets the value of the numK property.
 *
 * @param value
 *     allowed object is
 *     {@link Integer }
 *
 */
public void setNumK(Integer value) {
   this.numK = value;
}
/**
 * Gets the value of the numE property.
 *
 * @return
 *     possible object is
 *     {@link Integer }
 *
 */
public Integer getNumE() {
   return numE;
}
/**
 * Sets the value of the numE property.
 *
 * @param value
 *     allowed object is
 *     {@link Integer }
```

```
 *
 */
public void setNumE(Integer value) {
    this.numE = value;
}
/**
 * Gets the value of the tipo property.
 *
 * @return
 *     possible object is
 *     {@link Integer }
 *
 */
public Integer getTipo() {
    return tipo;
}
/**
 * Sets the value of the tipo property.
 *
 * @param value
 *     allowed object is
 *     {@link Integer }
 *
 */
public void setTipo(Integer value) {
    this.tipo = value;
}
/**
 * Gets the value of the numO property.
 *
 * @return
 *     possible object is
 *     {@link Integer }
 *
 */
public Integer getNumO() {
    return numO;
}
/**
 * Sets the value of the numO property.
 *
 * @param value
 *     allowed object is
 *     {@link Integer }
 *
 */
public void setNumO(Integer value) {
    this.numO = value;
}
/**
```

```
    * Gets the value of the iniciaDemanda property.
    *
    * @return
    *     possible object is
    *     {@link Integer }
    *
    */
   public Integer getIniciaDemanda() {
      return iniciaDemanda;
   }
   /**
    * Sets the value of the iniciaDemanda property.
    *
    * @param value
    *     allowed object is
    *     {@link Integer }
    *
    */
   public void setIniciaDemanda(Integer value) {
      this.iniciaDemanda = value;
   }
}
```

```
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v2.1-b02-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jax
b</a>
// Any modifications to this file will be lost upon
recompilation of the source schema.
// Generated on: 2010.04.15 at 11:52:41 PM CEST
//


package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for DemandaEs complex type.
 *
 * <p>The following schema fragment specifies the expected
content contained within this class.
 *
 * <pre>
 * &lt;complexType name="DemandaEs">
 *   &lt;complexContent>
 *     &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
```

```
 *       &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="e"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="DESite"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "DemandaEs")
public class DemandaEs {

    @XmlAttribute
    protected String i;
    @XmlAttribute
    protected String t;
    @XmlAttribute
    protected String e;
    @XmlAttribute(name = "DESite")
    protected Double deSite;

    /**
     * Gets the value of the i property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getI() {
        return i;
    }

    /**
     * Sets the value of the i property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setI(String value) {
        this.i = value;
    }

    /**
     * Gets the value of the t property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getT() {
        return t;
    }
```

```
/**
 * Sets the value of the t property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setT(String value) {
    this.t = value;
}

/**
 * Gets the value of the e property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getE() {
    return e;
}

/**
 * Sets the value of the e property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setE(String value) {
    this.e = value;
}

/**
 * Gets the value of the deSite property.
 *
 * @return
 *     possible object is
 *     {@link Double }
 *
 */
public Double getDESite() {
    return deSite;
}

/**
 * Sets the value of the deSite property.
 *
 * @param value
 *     allowed object is
 *     {@link Double }
 *
 */
public void setDESite(Double value) {
    this.deSite = value;
}
```

```
}

package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for DemandaRe complex type.
 *
 * <p>The following schema fragment specifies the expected
content contained within this class.
 *
 * <pre>
 * &lt;complexType name="DemandaRe">
 *   &lt;complexContent>
 *     &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="o"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="Dito"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "DemandaRe")
public class DemandaRe {

    @XmlAttribute
    protected String i;
    @XmlAttribute
    protected String t;
    @XmlAttribute
    protected String o;
    @XmlAttribute(name = "Dito")
    protected Double dito;

    /**
     * Gets the value of the i property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getI() {
        return i;
    }
```

```
/**
 * Sets the value of the i property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setI(String value) {
    this.i = value;
}

/**
 * Gets the value of the t property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getT() {
    return t;
}

/**
 * Sets the value of the t property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setT(String value) {
    this.t = value;
}

/**
 * Gets the value of the o property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getO() {
    return o;
}

/**
 * Sets the value of the o property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setO(String value) {
    this.o = value;
}

/**
```

```
     * Gets the value of the dito property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getDito() {
        return dito;
    }

    /**
     * Sets the value of the dito property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setDito(Double value) {
        this.dito = value;
    }

}

package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for DemBass complex type.
 *
 * <p>The following schema fragment specifies the expected
content contained within this class.
 *
 * <pre>
 * &lt;complexType name="DemBass">
 *   &lt;complexContent>
 *     &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="Tpi"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *       &lt;attribute name="Tqi"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *       &lt;attribute name="mEi"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *       &lt;attribute name="mRi"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *       &lt;attribute name="errori"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
```

```
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "DemBass")
public class DemBass {

    @XmlAttribute
    protected String i;
    @XmlAttribute(name = "Tpi")
    protected Double tpi;
    @XmlAttribute(name = "Tqi")
    protected Double tqi;
    @XmlAttribute
    protected Double mEi;
    @XmlAttribute
    protected Double mRi;
    @XmlAttribute
    protected Double errori;

    /**
     * Gets the value of the i property.
   *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getI() {
        return i;
    }

    /**
     * Sets the value of the i property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setI(String value) {
        this.i = value;
    }

    /**
     * Gets the value of the tpi property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getTpi() {
        return tpi;
    }

    /**
     * Sets the value of the tpi property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
```

```
     *
     */
    public void setTpi(Double value) {
        this.tpi = value;
    }

    /**
     * Gets the value of the tqi property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getTqi() {
        return tqi;
    }

    /**
     * Sets the value of the tqi property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setTqi(Double value) {
        this.tqi = value;
    }

    /**
     * Gets the value of the mEi property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getMEi() {
        return mEi;
    }

    /**
     * Sets the value of the mEi property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setMEi(Double value) {
        this.mEi = value;
    }

    /**
     * Gets the value of the mRi property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
```

```
     */
    public Double getMRi() {
        return mRi;
    }

    /**
     * Sets the value of the mRi property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setMRi(Double value) {
        this.mRi = value;
    }

    /**
     * Gets the value of the errori property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getErrori() {
        return errori;
    }

    /**
     * Sets the value of the errori property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setErrori(Double value) {
        this.errori = value;
    }

}
```

```
package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>Java class for InventarioIncial complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within this class.
 *
 * <pre>
 * &lt;complexType name="InventarioIncial">
```

```
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="INI"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "InventarioIncial")
public class InventarioIncial {

    @XmlAttribute
    protected String i;
    @XmlAttribute(name = "INI")
    protected Double ini;

    /**
     * Gets the value of the i property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getI() {
        return i;
    }
    /**
     * Sets the value of the i property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setI(String value) {
        this.i = value;
    }

    /**
     * Gets the value of the ini property.
     *
     * @return
     *     possible object is
```

```
     *     {@link Double }
     *
     */
    public Double getINI() {
        return ini;
    }
    /**
     * Sets the value of the ini property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setINI(Double value) {
        this.ini = value;
    }
}
```

```
package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for leadtime complex type.
 *
 * <p>The following schema fragment specifies the expected
content contained within this class.
 *
 * <pre>
 * &lt;complexType name="leadtime">
 *   &lt;complexContent>
 *     &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="k"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="LTk"
type="{http://www.w3.org/2001/XMLSchema}int" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "leadtime")
public class Leadtime {

    @XmlAttribute
    protected String k;
    @XmlAttribute(name = "LTk")
    protected Integer lTk;
```

```
    /**
     * Gets the value of the k property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getK() {
        return k;
    }

    /**
     * Sets the value of the k property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setK(String value) {
        this.k = value;
    }

    /**
     * Gets the value of the lTk property.
     *
     * @return
     *     possible object is
     *     {@link Integer }
     *
     */
    public Integer getLTk() {
        return lTk;
    }

    /**
     * Sets the value of the lTk property.
     *
     * @param value
     *     allowed object is
     *     {@link Integer }
     *
     */
    public void setLTk(Integer value) {
        this.lTk = value;
    }

}
```

```
package LectorXML;

import javax.xml.bind.annotation.XmlRegistry;


/**
 * This object contains factory methods for each
 * Java content interface and Java element interface
 * generated in the LectorXML package.
 * <p>An ObjectFactory allows you to programatically
 * construct new instances of the Java representation
 * for XML content. The Java representation of XML
 * content can consist of schema derived interfaces
 * and classes representing the binding of schema
 * type definitions, element declarations and model
 * groups.  Factory methods for each of these are
 * provided in this class.
 *
 */
@XmlRegistry
public class ObjectFactory {


    /**
     * Create a new ObjectFactory that can be used to create
new instances of schema derived classes for package: LectorXML
     *
     */
    public ObjectFactory() {
    }

    /**
     * Create an instance of {@link CosteInventory }
     *
     */
    public CosteInventory createCosteInventory() {
        return new CosteInventory();
    }

    /**
     * Create an instance of {@link PenalidadMAS }
     *
     */
    public PenalidadMAS createPenalidadMAS() {
        return new PenalidadMAS();
    }

    /**
     * Create an instance of {@link InventarioIncial }
     *
     */
    public InventarioIncial createInventarioIncial() {
        return new InventarioIncial();
    }

    /**
     * Create an instance of {@link PrecioProd }
     *
     */
    public PrecioProd createPrecioProd() {
        return new PrecioProd();
```

```
}

/**
 * Create an instance of {@link ActivaCompra }
 *
 */
public ActivaCompra createActivaCompra() {
    return new ActivaCompra();
}

/**
 * Create an instance of {@link Data }
 *
 */
public Data createData() {
    return new Data();
}

/**
 * Create an instance of {@link PrecioCompra }
 *
 */
public PrecioCompra createPrecioCompra() {
    return new PrecioCompra();
}

/**
 * Create an instance of {@link DemBass }
 *
 */
public DemBass createDemBass() {
    return new DemBass();
}

/**
 * Create an instance of {@link BOM }
 *
 */
public BOM createBOM() {
    return new BOM();
}

/**
 * Create an instance of {@link PenalidadMENOS }
 *
 */
public PenalidadMENOS createPenalidadMENOS() {
    return new PenalidadMENOS();
}

/**
 * Create an instance of {@link CosteBackOrder }
 *
 */
public CosteBackOrder createCosteBackOrder() {
    return new CosteBackOrder();
}

/**
 * Create an instance of {@link ProbabilidadEsc }
 *
```

```
 */
public ProbabilidadEsc createProbabilidadEsc() {
    return new ProbabilidadEsc();
}

/**
 * Create an instance of {@link DemandaEs }
 *
 */
public DemandaEs createDemandaEs() {
    return new DemandaEs();
}

/**
 * Create an instance of {@link DemandaRe }
 *
 */
public DemandaRe createDemandaRe() {
    return new DemandaRe();
}

/**
 * Create an instance of {@link CosCapacidad }
 *
 */
public CosCapacidad createCosCapacidad() {
    return new CosCapacidad();
}

/**
 * Create an instance of {@link RBOM }
 *
 */
public RBOM createRBOM() {
    return new RBOM();
}

/**
 * Create an instance of {@link Leadtime }
 *
 */
public Leadtime createLeadtime() {
    return new Leadtime();
}

/**
 * Create an instance of {@link CoefTecno }
 *
 */
public CoefTecno createCoefTecno() {
    return new CoefTecno();
}

/**
 * Create an instance of {@link ResidualInv }
 *
 */
public ResidualInv createResidualInv() {
    return new ResidualInv();
}
```

```
    /**
     * Create an instance of {@link UpperBounCapacidad }
     *
     */
    public UpperBounCapacidad createUpperBounCapacidad() {
        return new UpperBounCapacidad();
    }

    /**
     * Create an instance of {@link CosteStroke }
     *
     */
    public CosteStroke createCosteStroke() {
        return new CosteStroke();
    }

}
```

```
package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>Java class for PenalidadMAS complex type.
 *
 * <p>The following schema fragment specifies the expected content contained
within this class.
 *
 * <pre>
 * &lt;complexType name="PenalidadMAS">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="r"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="PNmasrt"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "PenalidadMAS")
public class PenalidadMAS {

    @XmlAttribute
```

```
protected String r;
@XmlAttribute
protected String t;
@XmlAttribute(name = "PNmasrt")
protected Double pNmasrt;

/**
 * Gets the value of the r property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getR() {
   return r;
}
/**
 * Sets the value of the r property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setR(String value) {
   this.r = value;
}
/**
 * Gets the value of the t property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getT() {
   return t;
}
/**
 * Sets the value of the t property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setT(String value) {
   this.t = value;
}
```

```
    /**
     * Gets the value of the pNmasrt property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getPNmasrt() {
        return pNmasrt;
    }
    /**
     * Sets the value of the pNmasrt property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setPNmasrt(Double value) {
        this.pNmasrt = value;
    }
}

package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>Java class for PenalidadMENOS complex type.
 *
 * <p>The following schema fragment specifies the expected content contained
within this class.
 *
 * <pre>
 * &lt;complexType name="PenalidadMENOS">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="r"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="PNmenosrt"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
```

```
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "PenalidadMENOS")
public class PenalidadMENOS {

    @XmlAttribute
    protected String r;
    @XmlAttribute
    protected String t;
    @XmlAttribute(name = "PNmenosrt")
    protected Double pNmenosrt;

    /**
     * Gets the value of the r property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getR() {
        return r;
    }
    /**
     * Sets the value of the r property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setR(String value) {
        this.r = value;
    }
    /**
     * Gets the value of the t property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getT() {
        return t;
    }
    /**
     * Sets the value of the t property.
```

```
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setT(String value) {
        this.t = value;
    }
    /**
     * Gets the value of the pNmenosrt property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getPNmenosrt() {
        return pNmenosrt;
    }
    /**
     * Sets the value of the pNmenosrt property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setPNmenosrt(Double value) {
        this.pNmenosrt = value;
    }
}

// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v2.1-b02-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source
schema.
// Generated on: 2010.04.15 at 11:52:41 PM CEST
//

package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

/**
```

```
 * <p>Java class for PrecioCompra complex type.
 *
 * <p>The following schema fragment specifies the expected content contained
within this class.
 *
 * <pre>
 * &lt;complexType name="PrecioCompra">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="PCit"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "PrecioCompra")
public class PrecioCompra {

    @XmlAttribute
    protected String i;
    @XmlAttribute
    protected String t;
    @XmlAttribute(name = "PCit")
    protected Double pCit;

    /**
     * Gets the value of the i property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getI() {
        return i;
    }
    /**
     * Sets the value of the i property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
```

```
     *
     */
    public void setI(String value) {
        this.i = value;
    }
    /**
     * Gets the value of the t property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getT() {
        return t;
    }
    /**
     * Sets the value of the t property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setT(String value) {
        this.t = value;
    }
    /**
     * Gets the value of the pCit property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getPCit() {
        return pCit;
    }
    /**
     * Sets the value of the pCit property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setPCit(Double value) {
        this.pCit = value;
    }
}
```

```
package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>Java class for PrecioProd complex type.
 *
 * <p>The following schema fragment specifies the expected content contained
within this class.
 *
 * <pre>
 * &lt;complexType name="PrecioProd">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="PVi"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "PrecioProd")
public class PrecioProd {

    @XmlAttribute
    protected String i;
    @XmlAttribute(name = "PVi")
    protected Double pVi;

    /**
     * Gets the value of the i property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getI() {
        return i;
    }
    /**
```

```
     * Sets the value of the i property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setI(String value) {
        this.i = value;
    }
    /**
     * Gets the value of the pVi property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getPVi() {
        return pVi;
   }
    /**
     * Sets the value of the pVi property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setPVi(Double value) {
        this.pVi = value;
    }
}

package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>Java class for ProbabilidadEsc complex type.
 *
 * <p>The following schema fragment specifies the expected content contained
within this class.
 *
 * <pre>
 * &lt;complexType name="ProbabilidadEsc">
 *   &lt;complexContent>
```

```
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="e"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="Pe"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "ProbabilidadEsc")
public class ProbabilidadEsc {

    @XmlAttribute
    protected String e;
    @XmlAttribute(name = "Pe")
    protected Double pe;

    /**
     * Gets the value of the e property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getE() {
        return e;
    }
    /**
     * Sets the value of the e property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setE(String value) {
        this.e = value;
    }
    /**
     * Gets the value of the pe property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
```

```
     */
    public Double getPe() {
        return pe;
    }
    /**
     * Sets the value of the pe property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setPe(Double value) {
        this.pe = value;
    }
}

package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>Java class for rBOM complex type.
 *
 * <p>The following schema fragment specifies the expected content contained
within this class.
 *
 * <pre>
 * &lt;complexType name="rBOM">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="k"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="Nik"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "rBOM")
public class RBOM {
```

```
@XmlAttribute
protected String i;
@XmlAttribute
protected String k;
@XmlAttribute(name = "Nik")
protected Double nik;

/**
 * Gets the value of the i property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getI() {
   return i;
}
/**
 * Sets the value of the i property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setI(String value) {
   this.i = value;
}
/**
 * Gets the value of the k property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getK() {
   return k;
}
/**
 * Sets the value of the k property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setK(String value) {
```

```
        this.k = value;
    }
    /**
     * Gets the value of the nik property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getNik() {
        return nik;
    }
    /**
     * Sets the value of the nik property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setNik(Double value) {
        this.nik = value;
    }
}

package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>Java class for ResidualInv complex type.
 *
 * <p>The following schema fragment specifies the expected content contained
within this class.
 *
 * <pre>
 * &lt;complexType name="ResidualInv">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="i"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="VRit"
type="{http://www.w3.org/2001/XMLSchema}double" />
 *     &lt;/restriction>
```

```
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "ResidualInv")
public class ResidualInv {

    @XmlAttribute
    protected String i;
    @XmlAttribute
    protected String t;
    @XmlAttribute(name = "VRit")
    protected Double vRit;

    /**
     * Gets the value of the i property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getI() {
        return i;
    }
    /**
     * Sets the value of the i property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setI(String value) {
        this.i = value;
    }
    /**
     * Gets the value of the t property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getT() {
        return t;
    }
```

```
    /**
     * Sets the value of the t property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setT(String value) {
        this.t = value;
    }
    /**
     * Gets the value of the vRit property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getVRit() {
        return vRit;
    }
    /**
     * Sets the value of the vRit property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setVRit(Double value) {
        this.vRit = value;
    }
}

package LectorXML;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>Java class for UpperBounCapacidad complex type.
 *
 * <p>The following schema fragment specifies the expected content contained
within this class.
 *
 * <pre>
 * &lt;complexType name="UpperBounCapacidad">
```

```
 * &lt;complexContent>
 * &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 * &lt;attribute name="r"
type="{http://www.w3.org/2001/XMLSchema}string" />
 * &lt;attribute name="t"
type="{http://www.w3.org/2001/XMLSchema}string" />
 * &lt;attribute name="KAPrt"
type="{http://www.w3.org/2001/XMLSchema}double" />
 * &lt;/restriction>
 * &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "UpperBounCapacidad")
public class UpperBounCapacidad {

    @XmlAttribute
    protected String r;
    @XmlAttribute
    protected String t;
    @XmlAttribute(name = "KAPrt")
    protected Double kaPrt;

    /**
     * Gets the value of the r property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getR() {
        return r;
    }
    /**
     * Sets the value of the r property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setR(String value) {
        this.r = value;
    }
    /**
     * Gets the value of the t property.
```

```
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getT() {
        return t;
    }
    /**
     * Sets the value of the t property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setT(String value) {
        this.t = value;
    }
    /**
     * Gets the value of the kaPrt property.
     *
     * @return
     *     possible object is
     *     {@link Double }
     *
     */
    public Double getKAPrt() {
        return kaPrt;
    }
    /**
     * Sets the value of the kaPrt property.
     *
     * @param value
     *     allowed object is
     *     {@link Double }
     *
     */
    public void setKAPrt(Double value) {
        this.kaPrt = value;
    }
}
```

Anexo 3

Datos de la demanda de cada escenario y sus probabilidades en el caso de Doble Proveedor (apartado 4.2.2. de Efecto de la incertidumbre y el número de escenarios)

En la tabla A3.1 se muestra el número de escenarios de cada alternativa analizada y sus probabilidades.

		Probabilidades						
Incertidumbre	Escenarios	1	2	3	4	5	6	7
Alta	3	0.40	0.30	0.30				
Alta	5	0.40	0.10	0.10	0.20	0.20		
Alta	7	0.30	0.10	0.10	0.20	0.20	0.05	0.05
Baja	3	0.40	0.30	0.30				
Baja	5	0.40	0.10	0.10	0.20	0.20		
Baja	7	0.30	0.10	0.10	0.20	0.20	0.05	0.05

Tabla A3.1: Escenarios y probabilidades.

En las siguientes tablas se muestra la demanda de los artículos para cada uno de los seis casos estudiados.

La tabla A3.2 muestra las demandas del caso de incertidumbre alta y tres escenarios.

SKU	Periodo	Escenario	Demanda
A	**1**	**1**	0
A	**2**	**1**	0
A	**3**	**1**	300
A	**4**	**1**	600
A	**5**	**1**	1000
A	**6**	**1**	500
B	**1**	**1**	0
B	**2**	**1**	0
B	**3**	**1**	0
B	**4**	**1**	0
B	**5**	**1**	0
B	**6**	**1**	0
C	**1**	**1**	0
C	**2**	**1**	0
C	**3**	**1**	0
C	**4**	**1**	0
C	**5**	**1**	0
C	**6**	**1**	0
A	**1**	**2**	0
A	**2**	**2**	0

A	**3**	**2**	650
A	**4**	**2**	1100
A	**5**	**2**	1950
A	**6**	**2**	600
B	**1**	**2**	0
B	**2**	**2**	0
B	**3**	**2**	0
B	**4**	**2**	0
B	**5**	**2**	0
B	**6**	**2**	0
C	**1**	**2**	0
C	**2**	**2**	0
C	**3**	**2**	0
C	**4**	**2**	0
C	**5**	**2**	0
C	**6**	**2**	0
A	**1**	**3**	0
A	**2**	**3**	0
A	**3**	**3**	20
A	**4**	**3**	140
A	**5**	**3**	300
A	**6**	**3**	20
B	**1**	**3**	0
B	**2**	**3**	0
B	**3**	**3**	0
B	**4**	**3**	0
B	**5**	**3**	0
B	**6**	**3**	0
C	**1**	**3**	0
C	**2**	**3**	0
C	**3**	**3**	0
C	**4**	**3**	0
C	**5**	**3**	0
C	**6**	**3**	0

Tabla A3.2: Demandas de incertidumbre alta y tres escenarios.

La tabla A3.3 muestra las demandas del caso de incertidumbre alta y cinco escenarios.

SKU	Periodo	Escenario	Demanda
A	**1**	**1**	0
A	**2**	**1**	0
A	**3**	**1**	300
A	**4**	**1**	600
A	**5**	**1**	1000
A	**6**	**1**	500
B	**1**	**1**	0
B	**2**	**1**	0
B	**3**	**1**	0
B	**4**	**1**	0
B	**5**	**1**	0
B	**6**	**1**	0
C	**1**	**1**	0
C	**2**	**1**	0
C	**3**	**1**	0
C	**4**	**1**	0
C	**5**	**1**	0
C	**6**	**1**	0
A	**1**	**2**	0
A	**2**	**2**	0
A	**3**	**2**	500
A	**4**	**2**	1340
A	**5**	**2**	2000
A	**6**	**2**	500
B	**1**	**2**	0
B	**2**	**2**	0
B	**3**	**2**	0
B	**4**	**2**	0
B	**5**	**2**	0
B	**6**	**2**	0
C	**1**	**2**	0
C	**2**	**2**	0
C	**3**	**2**	0
C	**4**	**2**	0
C	**5**	**2**	0
C	**6**	**2**	0
A	**1**	**3**	0
A	**2**	**3**	0
A	**3**	**3**	50
A	**4**	**3**	100
A	**5**	**3**	230
A	**6**	**3**	100
B	**1**	**3**	0

B	**2**	**3**	0
B	**3**	**3**	0
B	**4**	**3**	0
B	**5**	**3**	0
B	**6**	**3**	0
C	**1**	**3**	0
C	**2**	**3**	0
C	**3**	**3**	0
C	**4**	**3**	0
C	**5**	**3**	0
C	**6**	**3**	0
A	**1**	**4**	0
A	**2**	**4**	0
A	**3**	**4**	300
A	**4**	**4**	700
A	**5**	**4**	1500
A	**6**	**4**	380
B	**1**	**4**	0
B	**2**	**4**	0
B	**3**	**4**	0
B	**4**	**4**	0
B	**5**	**4**	0
B	**6**	**4**	0
C	**1**	**4**	0
C	**2**	**4**	0
C	**3**	**4**	0
C	**4**	**4**	0
C	**5**	**4**	0
C	**6**	**4**	0
A	**1**	**5**	0
A	**2**	**5**	0
A	**3**	**5**	210
A	**4**	**5**	550
A	**5**	**5**	950
A	**6**	**5**	210
B	**1**	**5**	0
B	**2**	**5**	0
B	**3**	**5**	0
B	**4**	**5**	0
B	**5**	**5**	0
B	**6**	**5**	0
C	**1**	**5**	0
C	**2**	**5**	0
C	**3**	**5**	0
C	**4**	**5**	0

C	**5**	**5**	0
C	**6**	**5**	0

Tabla A3.3: Demandas de incertidumbre alta y cinco escenarios.

La tabla A3.4 muestra las demandas del caso de incertidumbre alta y siete escenarios.

SKU	Periodo	Escenario	Demanda
A	**1**	**1**	0
A	**2**	**1**	0
A	**3**	**1**	300
A	**4**	**1**	600
A	**5**	**1**	1000
A	**6**	**1**	500
B	**1**	**1**	0
B	**2**	**1**	0
B	**3**	**1**	0
B	**4**	**1**	0
B	**5**	**1**	0
B	**6**	**1**	0
C	**1**	**1**	0
C	**2**	**1**	0
C	**3**	**1**	0
C	**4**	**1**	0
C	**5**	**1**	0
C	**6**	**1**	0
A	**1**	**2**	0
A	**2**	**2**	0
A	**3**	**2**	450
A	**4**	**2**	860
A	**5**	**2**	1600
A	**6**	**2**	450
B	**1**	**2**	0
B	**2**	**2**	0
B	**3**	**2**	0
B	**4**	**2**	0
B	**5**	**2**	0
B	**6**	**2**	0
C	**1**	**2**	0
C	**2**	**2**	0
C	**3**	**2**	0
C	**4**	**2**	0
C	**5**	**2**	0
C	**6**	**2**	0
A	**1**	**3**	0

A	**2**	**3**	0
A	**3**	**3**	120
A	**4**	**3**	450
A	**5**	**3**	750
A	**6**	**3**	120
B	**1**	**3**	0
B	**2**	**3**	0
B	**3**	**3**	0
B	**4**	**3**	0
B	**5**	**3**	0
B	**6**	**3**	0
C	**1**	**3**	0
C	**2**	**3**	0
C	**3**	**3**	0
C	**4**	**3**	0
C	**5**	**3**	0
C	**6**	**3**	0
A	**1**	**4**	0
A	**2**	**4**	0
A	**3**	**4**	300
A	**4**	**4**	700
A	**5**	**4**	1500
A	**6**	**4**	380
B	**1**	**4**	0
B	**2**	**4**	0
B	**3**	**4**	0
B	**4**	**4**	0
B	**5**	**4**	0
B	**6**	**4**	0
C	**1**	**4**	0
C	**2**	**4**	0
C	**3**	**4**	0
C	**4**	**4**	0
C	**5**	**4**	0
C	**6**	**4**	0
A	**1**	**5**	0
A	**2**	**5**	0
A	**3**	**5**	210
A	**4**	**5**	550
A	**5**	**5**	950
A	**6**	**5**	210
B	**1**	**5**	0
B	**2**	**5**	0
B	**3**	**5**	0
B	**4**	**5**	0

B	**5**	**5**	0
B	**6**	**5**	0
C	**1**	**5**	0
C	**2**	**5**	0
C	**3**	**5**	0
C	**4**	**5**	0
C	**5**	**5**	0
C	**6**	**5**	0
A	**1**	**6**	0
A	**2**	**6**	0
A	**3**	**6**	500
A	**4**	**6**	1340
A	**5**	**6**	2000
A	**6**	**6**	500
B	**1**	**6**	0
B	**2**	**6**	0
B	**3**	**6**	0
B	**4**	**6**	0
B	**5**	**6**	0
B	**6**	**6**	0
C	**1**	**6**	0
C	**2**	**6**	0
C	**3**	**6**	0
C	**4**	**6**	0
C	**5**	**6**	0
C	**6**	**6**	0
A	**1**	**7**	0
A	**2**	**7**	0
A	**3**	**7**	50
A	**4**	**7**	100
A	**5**	**7**	230
A	**6**	**7**	100
B	**1**	**7**	0
B	**2**	**7**	0
B	**3**	**7**	0
B	**4**	**7**	0
B	**5**	**7**	0
B	**6**	**7**	0
C	**1**	**7**	0
C	**2**	**7**	0
C	**3**	**7**	0
C	**4**	**7**	0
C	**5**	**7**	0
C	**6**	**7**	0

Tabla A3.4 Demandas de incertidumbre alta y siete escenarios.

La tabla A3.5 muestra las demandas del caso de incertidumbre baja y tres escenarios.

SKU	Periodo	Escenario	Demanda
A	**1**	**1**	0
A	**2**	**1**	0
A	**3**	**1**	300
A	**4**	**1**	600
A	**5**	**1**	1000
A	**6**	**1**	500
B	**1**	**1**	0
B	**2**	**1**	0
B	**3**	**1**	0
B	**4**	**1**	0
B	**5**	**1**	0
B	**6**	**1**	0
C	**1**	**1**	0
C	**2**	**1**	0
C	**3**	**1**	0
C	**4**	**1**	0
C	**5**	**1**	0
C	**6**	**1**	0
A	**1**	**2**	0
A	**2**	**2**	0
A	**3**	**2**	450
A	**4**	**2**	860
A	**5**	**2**	1600
A	**6**	**2**	450
B	**1**	**2**	0
B	**2**	**2**	0
B	**3**	**2**	0
B	**4**	**2**	0
B	**5**	**2**	0
B	**6**	**2**	0
C	**1**	**2**	0
C	**2**	**2**	0
C	**3**	**2**	0
C	**4**	**2**	0
C	**5**	**2**	0
C	**6**	**2**	0
A	**1**	**3**	0
A	**2**	**3**	0
A	**3**	**3**	120
A	**4**	**3**	450
A	**5**	**3**	750

A	**6**	**3**	120
B	**1**	**3**	0
B	**2**	**3**	0
B	**3**	**3**	0
B	**4**	**3**	0
B	**5**	**3**	0
B	**6**	**3**	0
C	**1**	**3**	0
C	**2**	**3**	0
C	**3**	**3**	0
C	**4**	**3**	0
C	**5**	**3**	0
C	**6**	**3**	0

Tabla A3.5: Demandas de incertidumbre baja y tres escenarios.

La tabla A3.6 muestra las demandas del caso de incertidumbre baja y cinco escenarios.

SKU	Periodo	Escenario	Demanda
A	**1**	**1**	0
A	**2**	**1**	0
A	**3**	**1**	300
A	**4**	**1**	600
A	**5**	**1**	1000
A	**6**	**1**	500
B	**1**	**1**	0
B	**2**	**1**	0
B	**3**	**1**	0
B	**4**	**1**	0
B	**5**	**1**	0
B	**6**	**1**	0
C	**1**	**1**	0
C	**2**	**1**	0
C	**3**	**1**	0
C	**4**	**1**	0
C	**5**	**1**	0
C	**6**	**1**	0
A	**1**	**2**	0
A	**2**	**2**	0
A	**3**	**2**	450
A	**4**	**2**	860
A	**5**	**2**	1600
A	**6**	**2**	450
B	**1**	**2**	0

B	**2**	**2**	0
B	**3**	**2**	0
B	**4**	**2**	0
B	**5**	**2**	0
B	**6**	**2**	0
C	**1**	**2**	0
C	**2**	**2**	0
C	**3**	**2**	0
C	**4**	**2**	0
C	**5**	**2**	0
C	**6**	**2**	0
A	**1**	**3**	0
A	**2**	**3**	0
A	**3**	**3**	120
A	**4**	**3**	450
A	**5**	**3**	750
A	**6**	**3**	120
B	**1**	**3**	0
B	**2**	**3**	0
B	**3**	**3**	0
B	**4**	**3**	0
B	**5**	**3**	0
B	**6**	**3**	0
C	**1**	**3**	0
C	**2**	**3**	0
C	**3**	**3**	0
C	**4**	**3**	0
C	**5**	**3**	0
C	**6**	**3**	0
A	**1**	**4**	0
A	**2**	**4**	0
A	**3**	**4**	300
A	**4**	**4**	700
A	**5**	**4**	1500
A	**6**	**4**	380
B	**1**	**4**	0
B	**2**	**4**	0
B	**3**	**4**	0
B	**4**	**4**	0
B	**5**	**4**	0
B	**6**	**4**	0
C	**1**	**4**	0
C	**2**	**4**	0
C	**3**	**4**	0
C	**4**	**4**	0

C	**5**	**4**	0
C	**6**	**4**	0
A	**1**	**5**	0
A	**2**	**5**	0
A	**3**	**5**	210
A	**4**	**5**	550
A	**5**	**5**	950
A	**6**	**5**	210
B	**1**	**5**	0
B	**2**	**5**	0
B	**3**	**5**	0
B	**4**	**5**	0
B	**5**	**5**	0
B	**6**	**5**	0
C	**1**	**5**	0
C	**2**	**5**	0
C	**3**	**5**	0
C	**4**	**5**	0
C	**5**	**5**	0
C	**6**	**5**	0

Tabla A3.6: Demandas de incertidumbre baja y cinco escenarios.

La tabla A3.7 muestra las demandas del caso de incertidumbre baja y siete escenarios.

SKU	Periodo	Escenario	Demanda
A	**1**	**1**	0
A	**2**	**1**	0
A	**3**	**1**	300
A	**4**	**1**	600
A	**5**	**1**	1000
A	**6**	**1**	500
B	**1**	**1**	0
B	**2**	**1**	0
B	**3**	**1**	0
B	**4**	**1**	0
B	**5**	**1**	0
B	**6**	**1**	0
C	**1**	**1**	0
C	**2**	**1**	0
C	**3**	**1**	0
C	**4**	**1**	0
C	**5**	**1**	0
C	**6**	**1**	0
A	**1**	**2**	0

A	**2**	**2**	0
A	**3**	**2**	400
A	**4**	**2**	800
A	**5**	**2**	1600
A	**6**	**2**	400
B	**1**	**2**	0
B	**2**	**2**	0
B	**3**	**2**	0
B	**4**	**2**	0
B	**5**	**2**	0
B	**6**	**2**	0
C	**1**	**2**	0
C	**2**	**2**	0
C	**3**	**2**	0
C	**4**	**2**	0
C	**5**	**2**	0
C	**6**	**2**	0
A	**1**	**3**	0
A	**2**	**3**	0
A	**3**	**3**	100
A	**4**	**3**	420
A	**5**	**3**	730
A	**6**	**3**	100
B	**1**	**3**	0
B	**2**	**3**	0
B	**3**	**3**	0
B	**4**	**3**	0
B	**5**	**3**	0
B	**6**	**3**	0
C	**1**	**3**	0
C	**2**	**3**	0
C	**3**	**3**	0
C	**4**	**3**	0
C	**5**	**3**	0
C	**6**	**3**	0
A	**1**	**4**	0
A	**2**	**4**	0
A	**3**	**4**	300
A	**4**	**4**	700
A	**5**	**4**	1500
A	**6**	**4**	380
B	**1**	**4**	0
B	**2**	**4**	0
B	**3**	**4**	0
B	**4**	**4**	0

B	**5**	**4**	0
B	**6**	**4**	0
C	**1**	**4**	0
C	**2**	**4**	0
C	**3**	**4**	0
C	**4**	**4**	0
C	**5**	**4**	0
C	**6**	**4**	0
A	**1**	**5**	0
A	**2**	**5**	0
A	**3**	**5**	210
A	**4**	**5**	550
A	**5**	**5**	950
A	**6**	**5**	210
B	**1**	**5**	0
B	**2**	**5**	0
B	**3**	**5**	0
B	**4**	**5**	0
B	**5**	**5**	0
B	**6**	**5**	0
C	**1**	**5**	0
C	**2**	**5**	0
C	**3**	**5**	0
C	**4**	**5**	0
C	**5**	**5**	0
C	**6**	**5**	0
A	**1**	**6**	0
A	**2**	**6**	0
A	**3**	**6**	450
A	**4**	**6**	860
A	**5**	**6**	1600
A	**6**	**6**	450
B	**1**	**6**	0
B	**2**	**6**	0
B	**3**	**6**	0
B	**4**	**6**	0
B	**5**	**6**	0
B	**6**	**6**	0
C	**1**	**6**	0
C	**2**	**6**	0
C	**3**	**6**	0
C	**4**	**6**	0
C	**5**	**6**	0
C	**6**	**6**	0
A	**1**	**7**	0

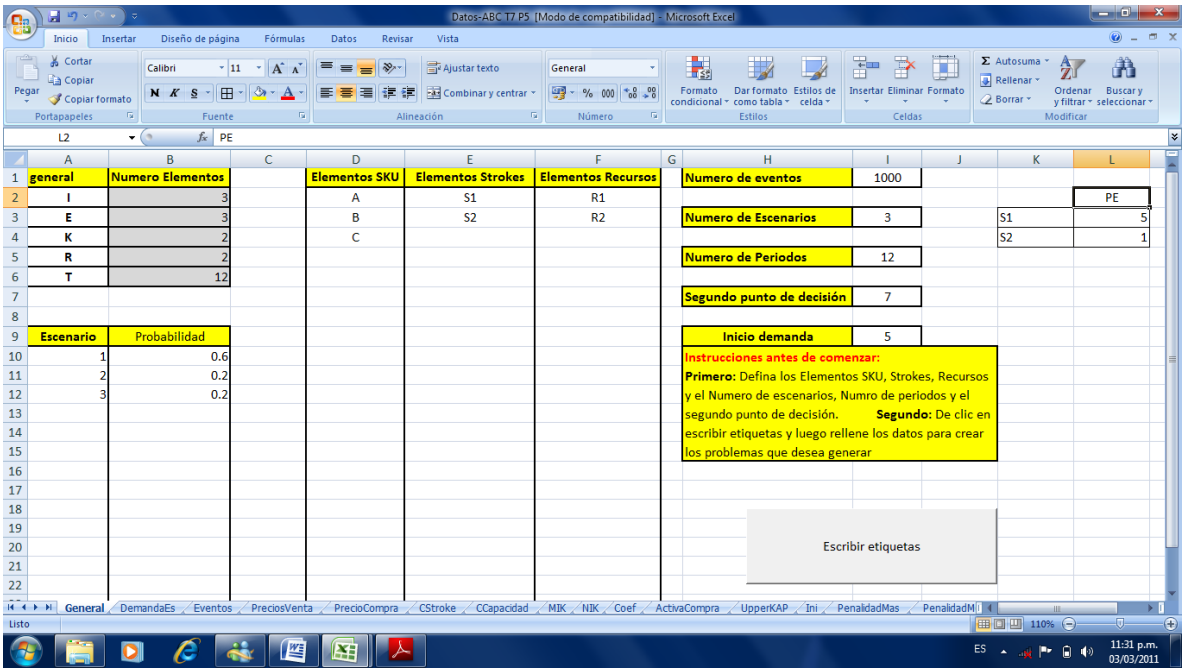
A	**2**	**7**	0
A	**3**	**7**	120
A	**4**	**7**	450
A	**5**	**7**	750
A	**6**	**7**	120
B	**1**	**7**	0
B	**2**	**7**	0
B	**3**	**7**	0
B	**4**	**7**	0
B	**5**	**7**	0
B	**6**	**7**	0
C	**1**	**7**	0
C	**2**	**7**	0
C	**3**	**7**	0
C	**4**	**7**	0
C	**5**	**7**	0
C	**6**	**7**	0

Tabla A3.7: Demandas de incertidumbre baja y siete escenarios.

Anexo 4

Datos de los problemas con dos proveedores para distintos plazos de entrega del proveedor lejano

En las siguientes tablas se presentan los datos de los problemas con dos proveedores, inicio de la demanda en el periodo 5 y segundo punto de decisión en el periodo 7, y plazo de entrega del proveedor lejano de 5 periodos. En la Tabla A4.1. se muestra la hoja de Excel de ingreso de datos para la resolución de los distintos casos experimentados. Se puede apreciar en la parte inferior que hay distintas páginas: "General", "DemandaEs" (Demanda Esperada), "Eventos", "PrecioVenta", etc., y que en esta tabla se está mostrando la hoja "General".



general	Numero Elementos
I	3
E	3
K	2
R	2
T	12

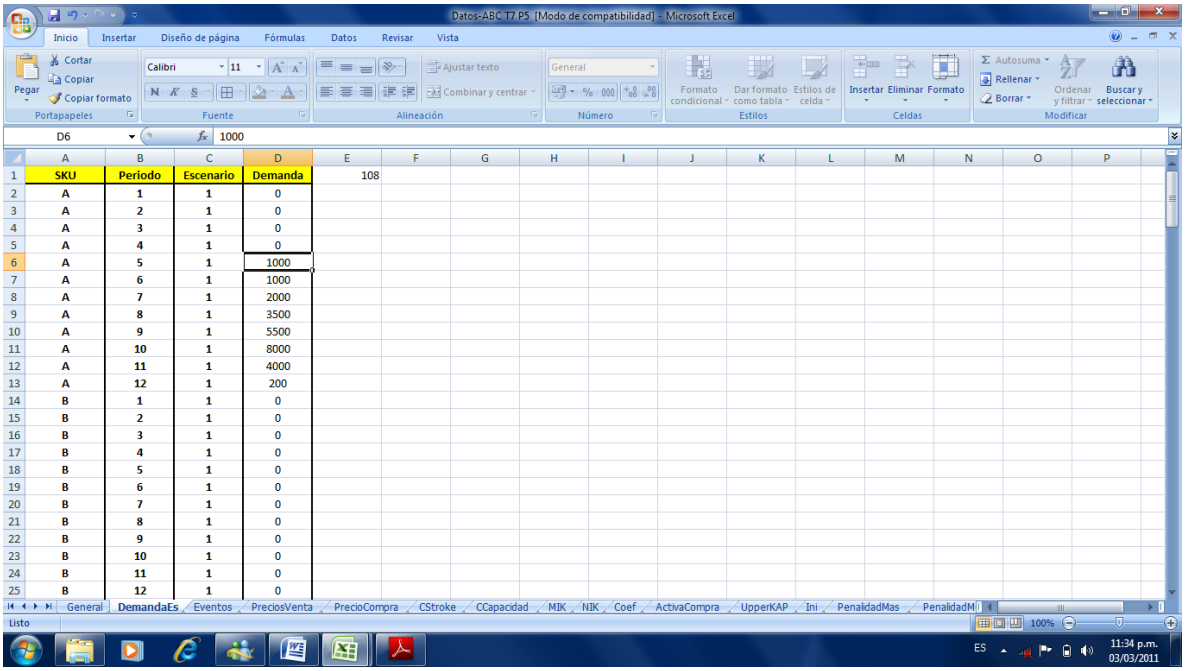
Elementos SKU	Elementos Strokes	Elementos Recursos
A	S1	R1
B	S2	R2
C		

Escenario	Probabilidad
1	0.6
2	0.2
3	0.2

	PE
S1	5
S2	1

Tabla A4.1. Hoja de Excel con datos de ingreso de cada problema estudiado.

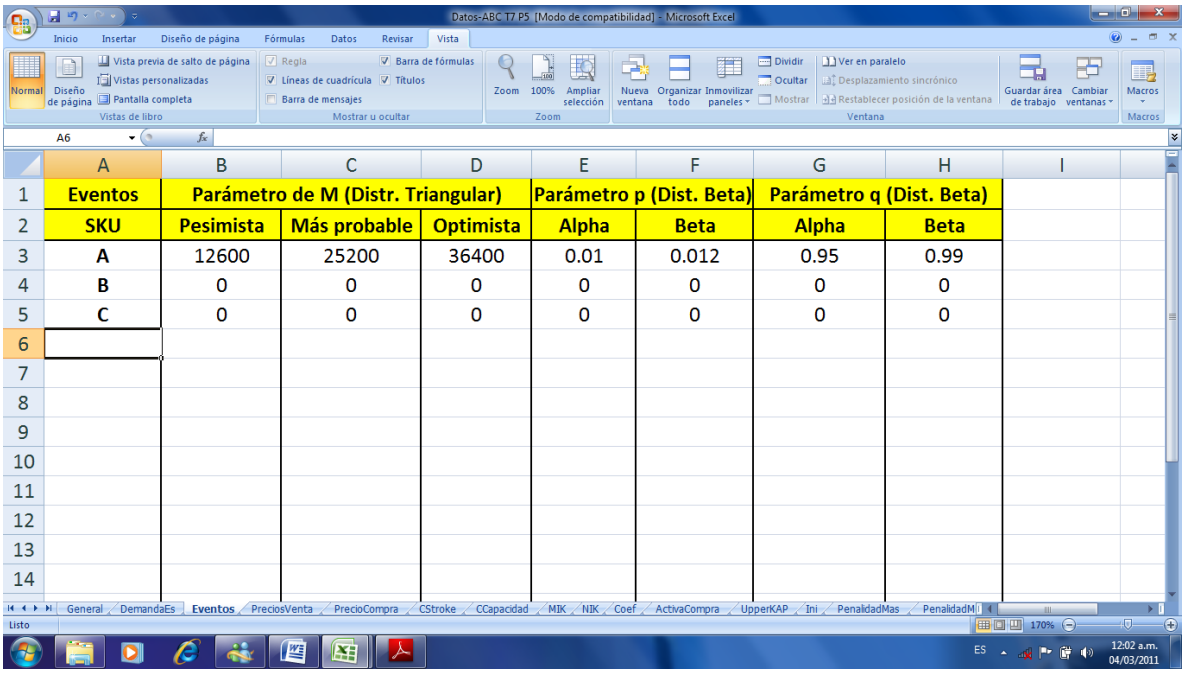
En la Tabla A4.2. se muestra parte de la Demanda correspondiente al primer escenario. Sólo el ítem A tiene demanda prevista. En la hoja de Excel está la información completa de demanda de los tres escenarios considerados. Esta información es como la mostrada en el Anexo 2.



SKU	Periodo	Escenario	Demanda
A	1	1	0
A	2	1	0
A	3	1	0
A	4	1	0
A	5	1	1000
A	6	1	1000
A	7	1	2000
A	8	1	3500
A	9	1	5500
A	10	1	8000
A	11	1	4000
A	12	1	200
B	1	1	0
B	2	1	0
B	3	1	0
B	4	1	0
B	5	1	0
B	6	1	0
B	7	1	0
B	8	1	0
B	9	1	0
B	10	1	0
B	11	1	0
B	12	1	0

Tabla A4.2. Página de los datos de Demanda Esperada para cada ítem (SKU) y para cada periodo y escenario.

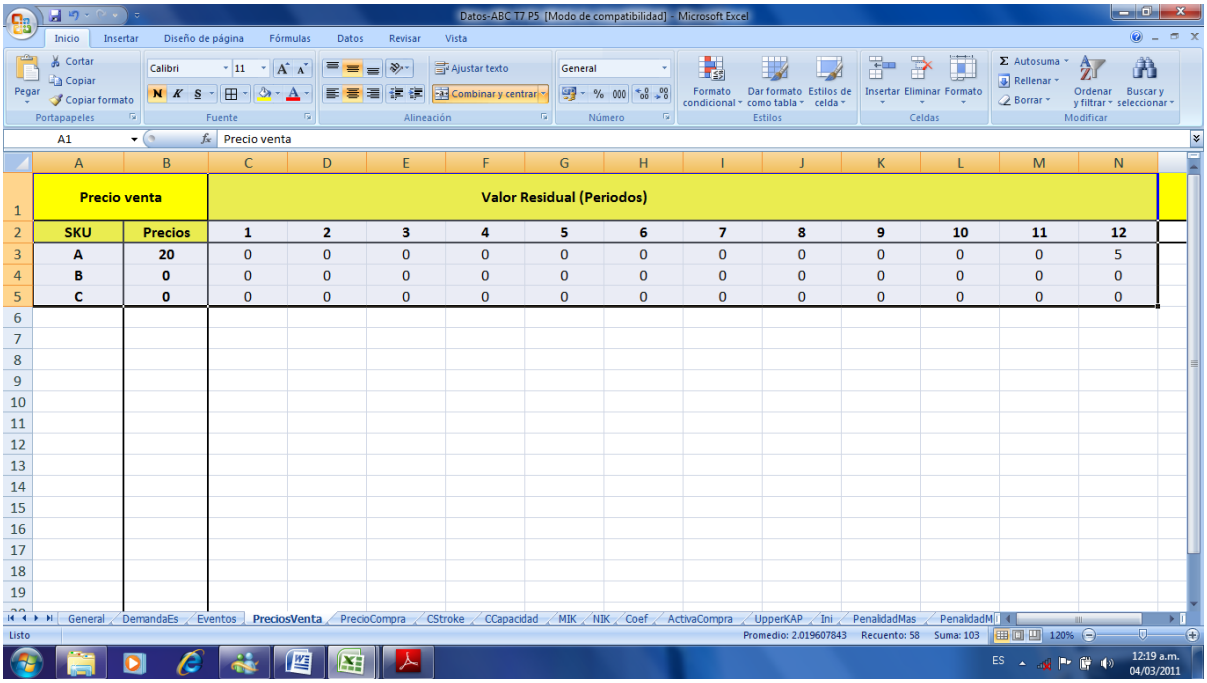
En la Tabla A4.3. se muestra la página de Eventos. En esta página se coloca la información necesaria para simular la demanda con el método de Bass.



Eventos	Parámetro de M (Distr. Triangular)			Parámetro p (Dist. Beta)		Parámetro q (Dist. Beta)	
SKU	Pesimista	Más probable	Optimista	Alpha	Beta	Alpha	Beta
A	12600	25200	36400	0.01	0.012	0.95	0.99
B	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0

Tabla A4.3. Página de los datos de Eventos para cada ítem (SKU).

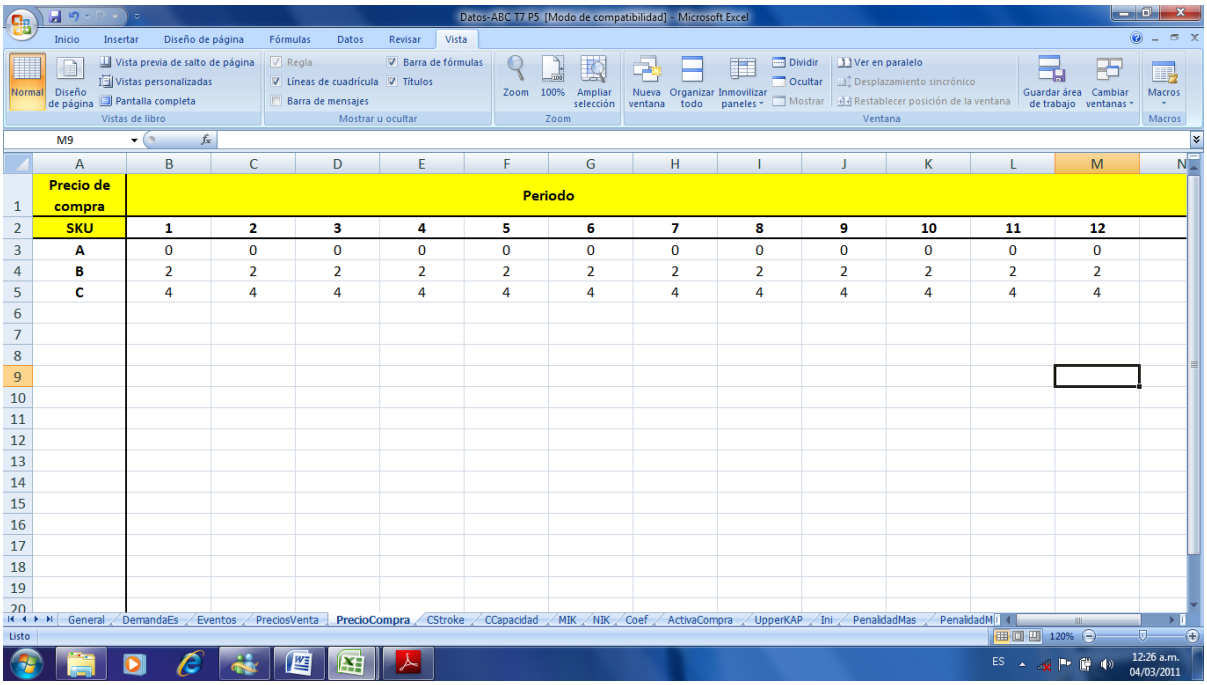
En la Tabla A4.4. se muestra la página de Precio de Venta (sólo el ítem A tiene precio de venta igual a 20 €/unidad) y el Valor Residual (sólo el ítem A tiene valor residual igual a 5 €/unidad en el último periodo).



Precio venta		Valor Residual (Periodos)											
SKU	Precios	1	2	3	4	5	6	7	8	9	10	11	12
A	20	0	0	0	0	0	0	0	0	0	0	0	5
B	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla A4.4. Página de los datos de Precio de venta para cada ítem (SKU).

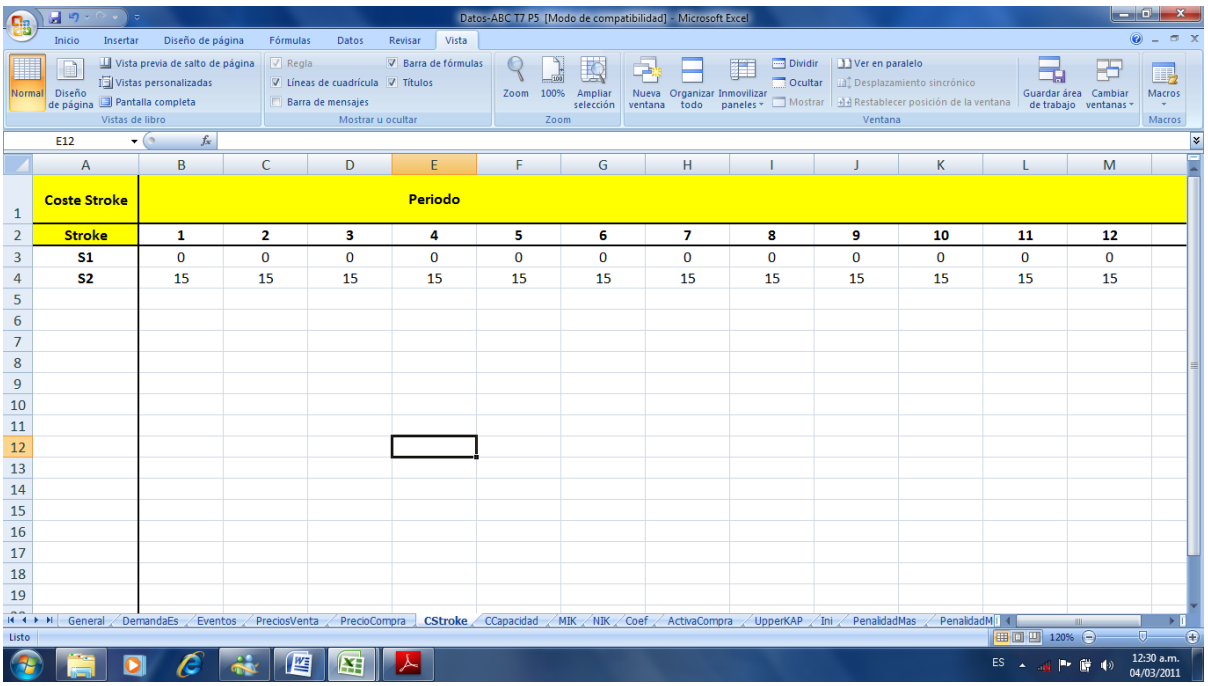
En la Tabla A4.5. se muestra la página de Precios de Compra de las materias primas para cada proveedor. La materia prima B corresponde al proveedor lejano y la materia prima C corresponde al proveedor local. Sus costes se mantienen constantes a los largo de los 12 periodos.



Precio de compra	Periodo											
SKU	1	2	3	4	5	6	7	8	9	10	11	12
A	0	0	0	0	0	0	0	0	0	0	0	0
B	2	2	2	2	2	2	2	2	2	2	2	2
C	4	4	4	4	4	4	4	4	4	4	4	4

Tabla A4.5. Página de los Precios de Compra de las materias primas.

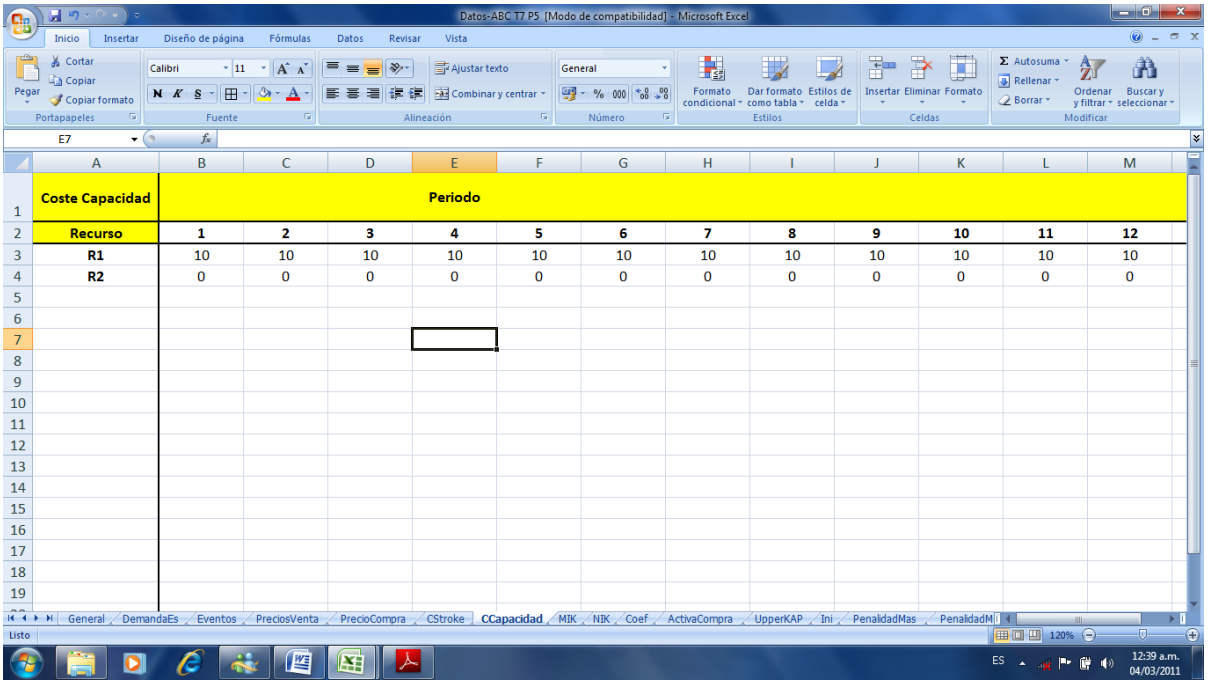
En la Tabla A4.6. se muestra la página de Coste de los *Strokes*. En esta página se coloca la información del coste unitario de los *strokes*. El *stroke* S1 corresponde al proveedor lejano y el *stroke* S2 al proveedor local. El coste está en €/unidad de producto terminado.



Coste Stroke	Periodo											
Stroke	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
S1	0	0	0	0	0	0	0	0	0	0	0	0
S2	15	15	15	15	15	15	15	15	15	15	15	15

Tabla A4.6. Página de los costes de los *strokes*.

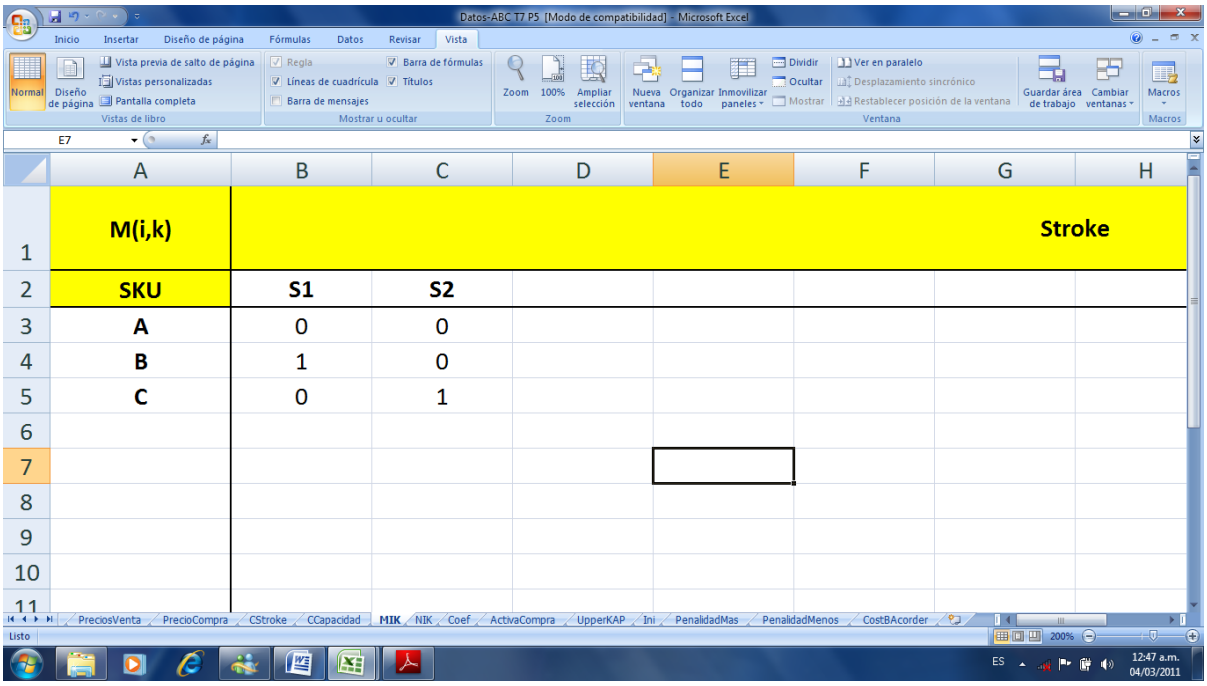
En la Tabla A4.7. se muestra la página de Costes de Capacidad. En esta página se colocan los costes de los recursos por unidad de producto en cada periodo. El recurso R1 corresponde al proveedor lejano y el R2 al proveedor local. En esta tabla y la anterior se evidencia que se contrata la capacidad en el proveedor lejano y se contrata la producción en el proveedor local.



Coste Capacidad	Periodo											
Recurso	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
R1	10	10	10	10	10	10	10	10	10	10	10	10
R2	0	0	0	0	0	0	0	0	0	0	0	0

Tabla A4.7. Página de los costes de los recursos.

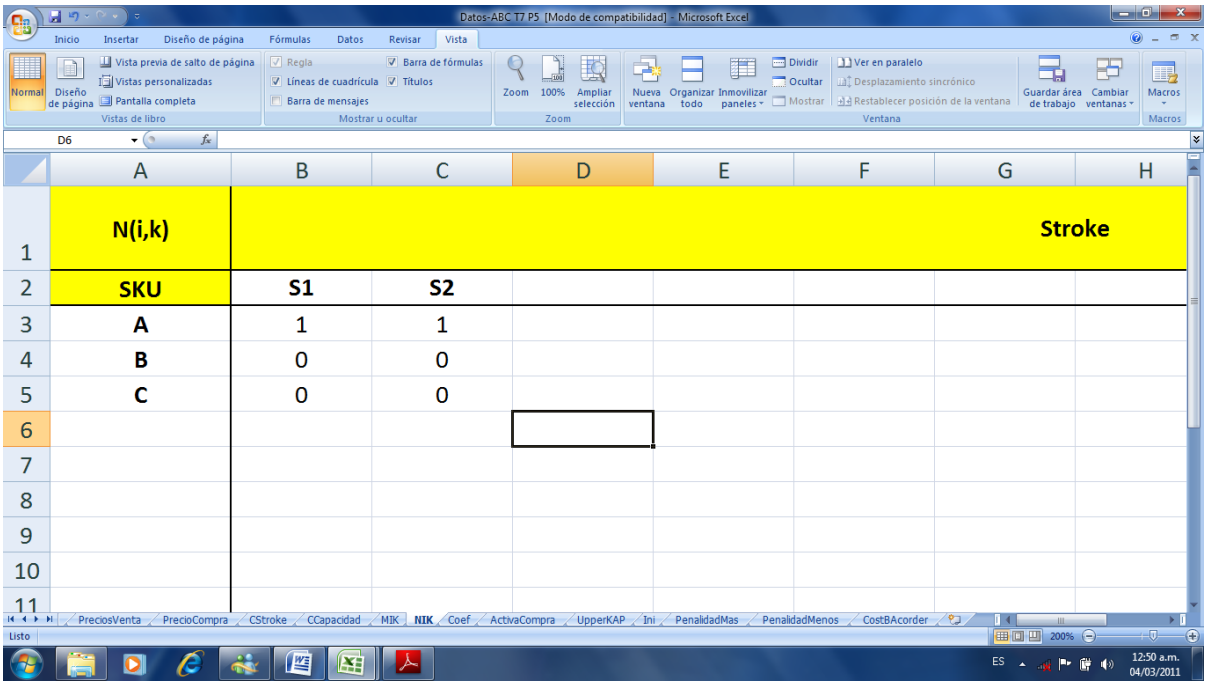
En la Tabla A4.8. se muestra la página de materiales que consume cada *stroke*. En esta página se ve que el *stroke* S1 consume una unidad de B y el *stroke* S2 consume una unidad de C.



M(i,k)		Stroke
SKU	S1	S2
A	0	0
B	1	0
C	0	1

Tabla A4.8. Página de los consumos de los *strokes*.

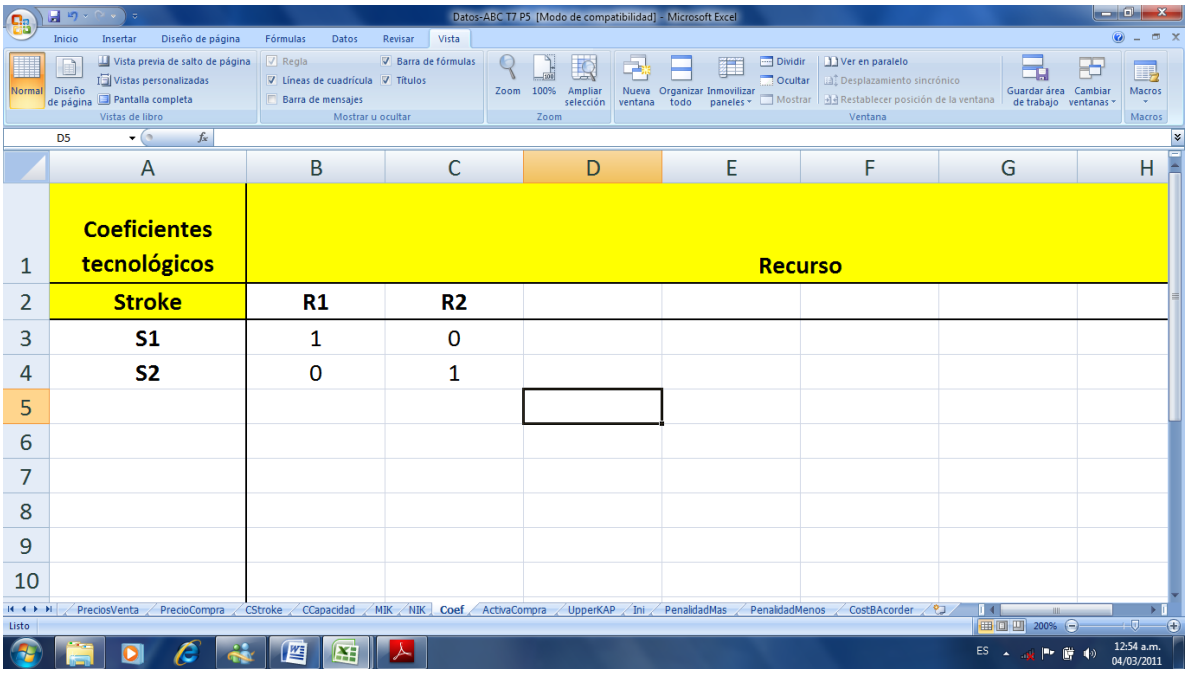
En la Tabla A4.9. se muestra la página de productos o salidas de los *strokes*. En esta página se ve que los dos *strokes*, S1 y S2, producen el ítem A.



N(i,k)		Stroke
SKU	S1	S2
A	1	1
B	0	0
C	0	0

Tabla A4.9. Página de los productos de los *strokes*.

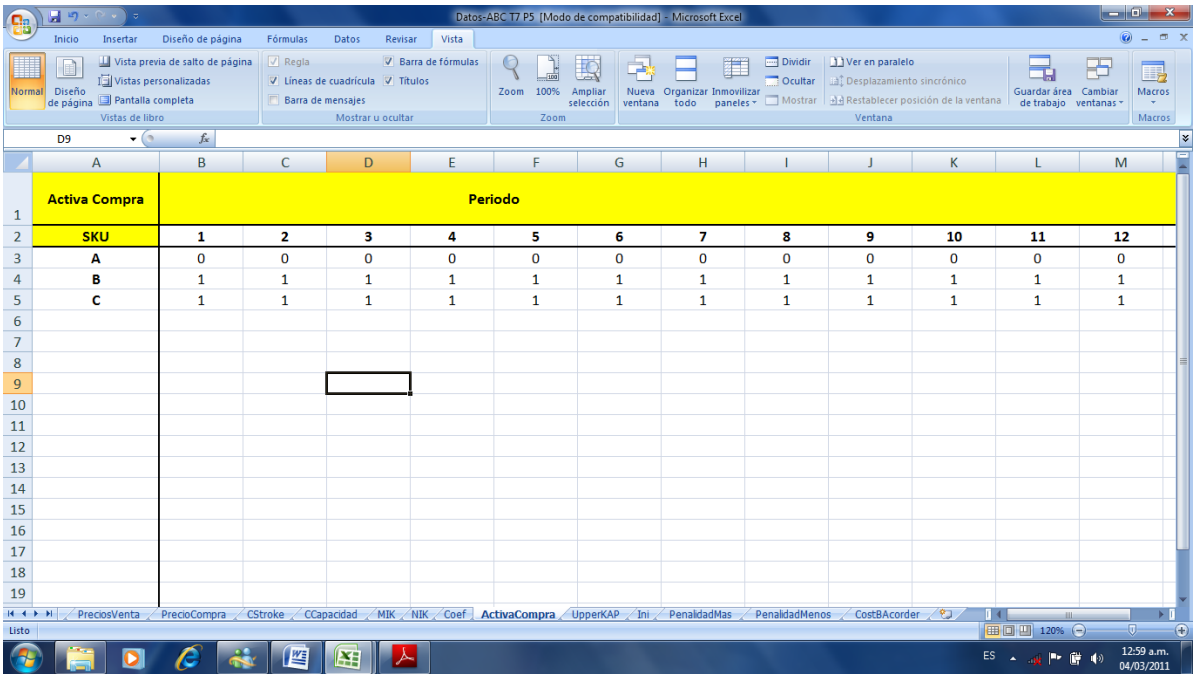
En la Tabla A4.10. se muestra la página de los Coeficientes tecnológicos que vinculan los recursos a cada *stroke*. En esta página ve que el *stroke* S1 consume una unidad del recurso R1, y que el S2 consume una unidad del R2.



	A	B	C	D	E	F	G	H
1	**Coeficientes tecnológicos**	**Recurso**						
2	**Stroke**	**R1**	**R2**					
3	**S1**	1	0					
4	**S2**	0	1					
5								
6								
7								
8								
9								
10								

Tabla A4.10. Página de los Coeficientes tecnológicos.

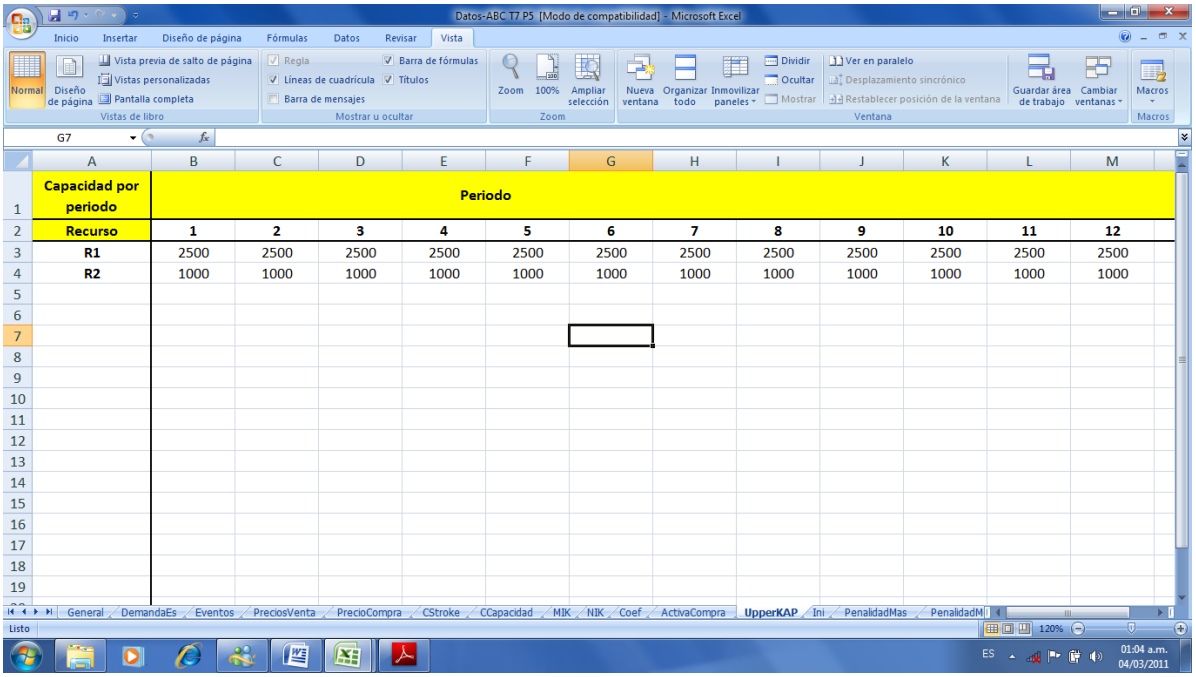
En la Tabla A4.11. se muestra la página "ActivaCompra". En esta página se colocan los coeficientes (0,1) que indican si el ítem se compra o no.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Activa Compra**	**Periodo**											
2	**SKU**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
3	**A**	0	0	0	0	0	0	0	0	0	0	0	0
4	**B**	1	1	1	1	1	1	1	1	1	1	1	1
5	**C**	1	1	1	1	1	1	1	1	1	1	1	1
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													

Tabla A4.11. Página de los coeficientes de compra.

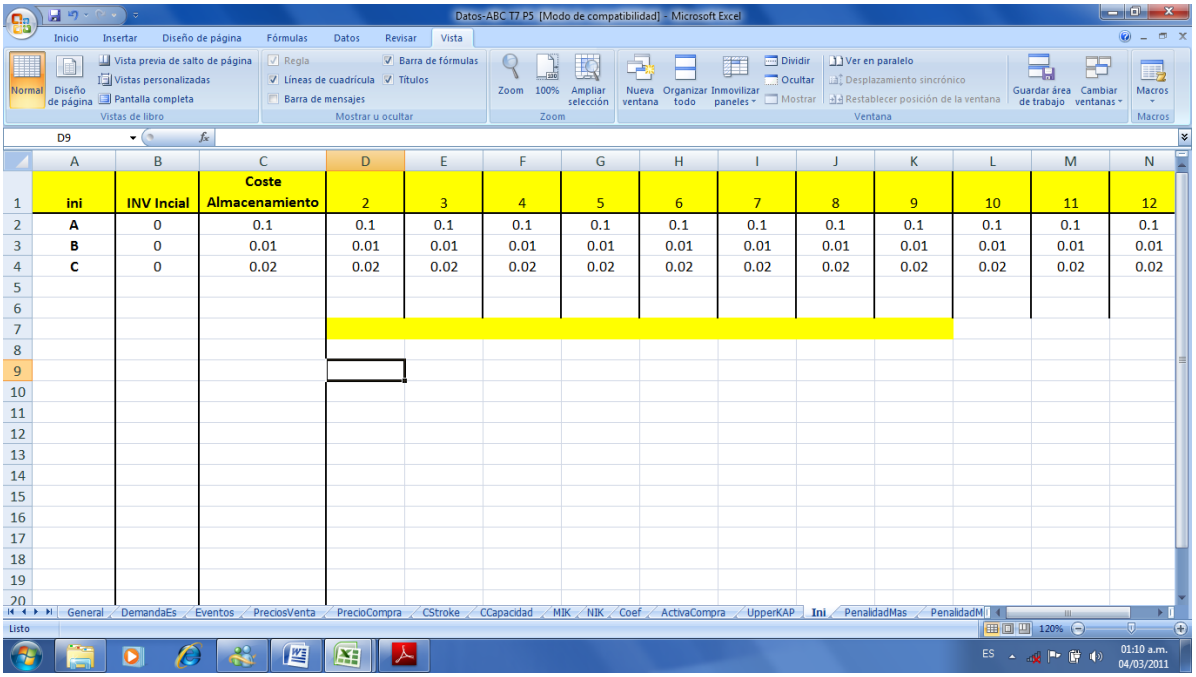
En la Tabla A4.12. se muestra la página de Capacidad por periodo. En esta página se coloca la capacidad de cada recurso en número de productos fabricados por periodo.



Capacidad por periodo	Periodo											
Recurso	1	2	3	4	5	6	7	8	9	10	11	12
R1	2500	2500	2500	2500	2500	2500	2500	2500	2500	2500	2500	2500
R2	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Tabla A4.12. Página de las capacidades de los recursos.

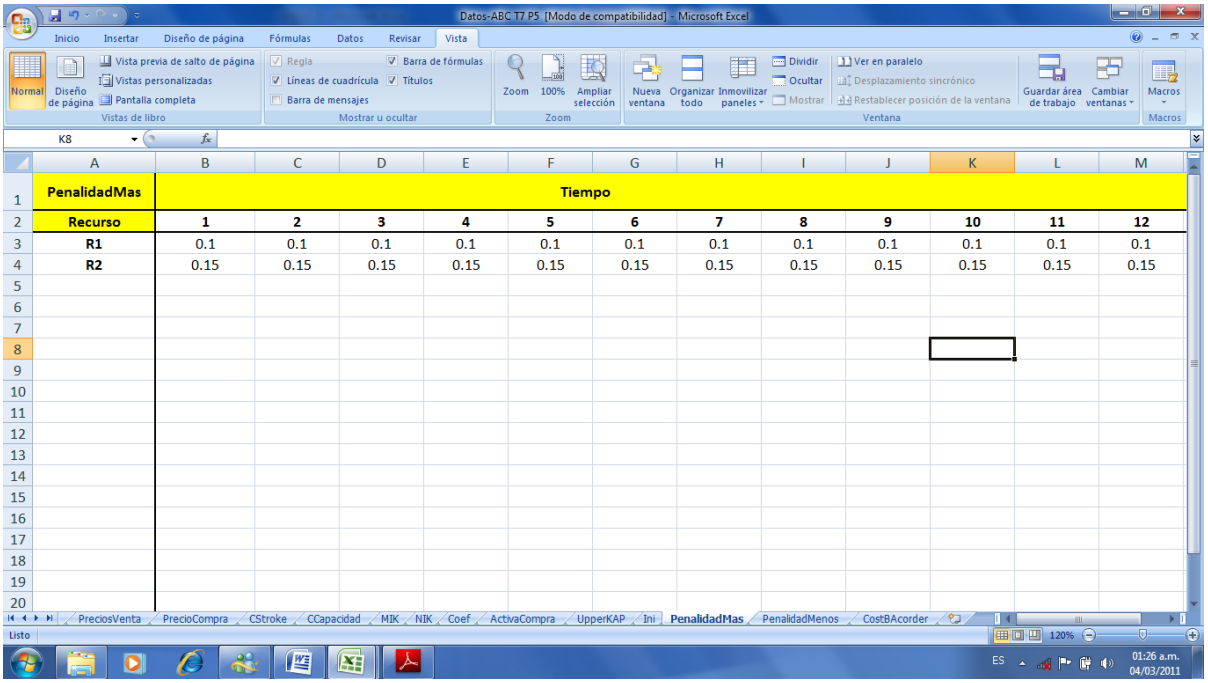
En la Tabla A4.13. se muestra la página "Ini" de Inventario Inicial y coste de almacenamiento. En esta página se coloca el inventario inicial de cada ítem y el coste unitario de almacenamiento, el cual puede ser diferente en cada periodo.



ini	INV Incial	Coste Almacenamiento	2	3	4	5	6	7	8	9	10	11	12
A	0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
B	0	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
C	0	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02

Tabla A4.13. Página de los costes unitarios de inventario.

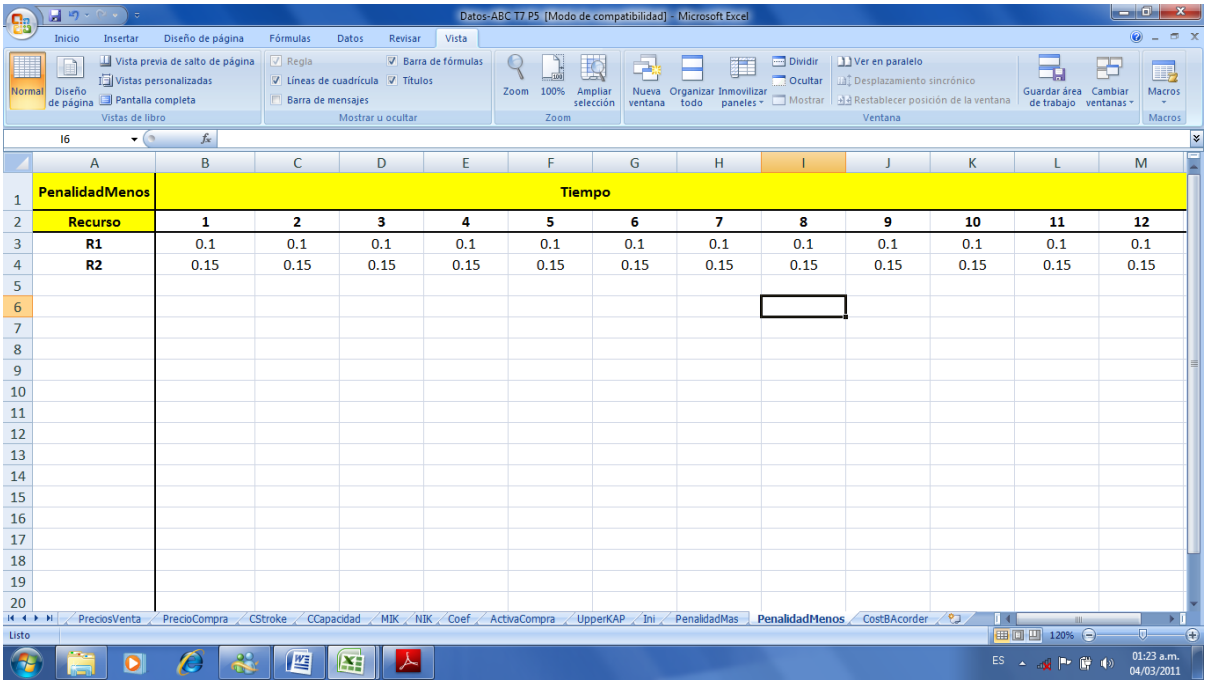
En la Tabla A4.14. se muestra la página de "PenalidadMas". En esta página se coloca el sobrecoste debido a usar más capacidad de la contratada en el proveedor lejano o más capacidad de la existente en el proveedor local (llamada capacidad máxima del proveedor local). Este coste está en €/unidad del recurso considerado.



PenalidadMas	Tiempo											
Recurso	1	2	3	4	5	6	7	8	9	10	11	12
R1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
R2	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15

Tabla A4.14. Página de los sobrecostes de las capacidades.

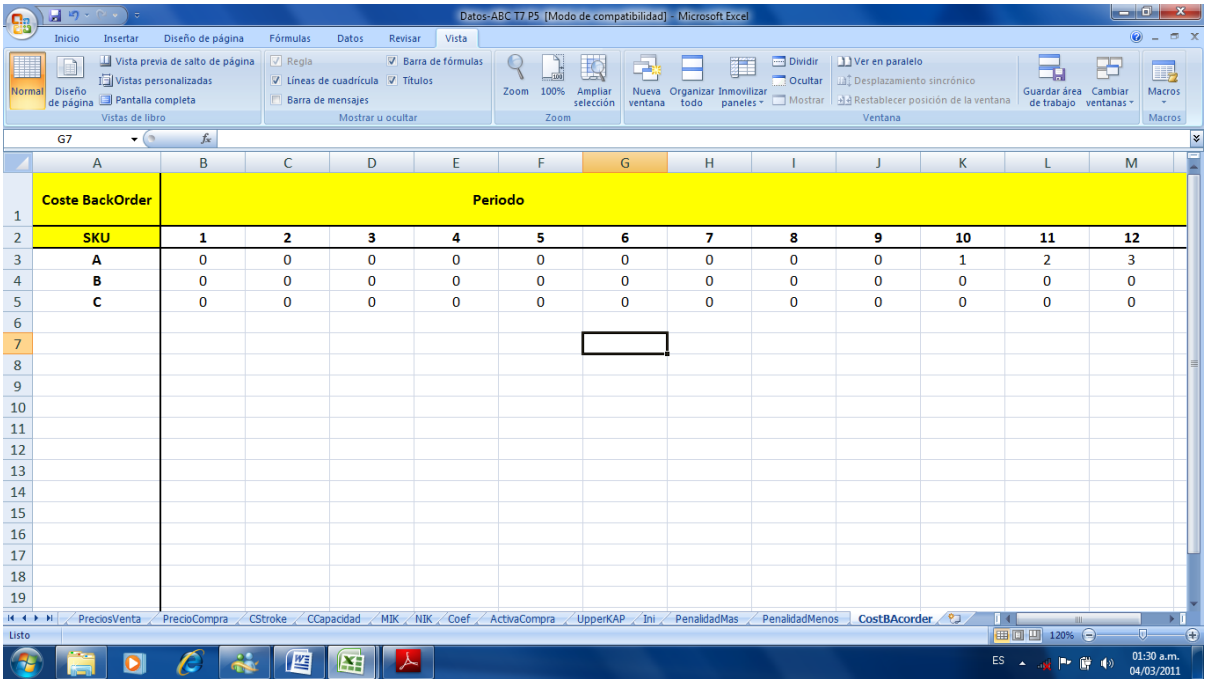
En la Tabla A4.15. se muestra la página de "PenalidadMenos". En esta página se coloca el sobrecoste por no usar toda la capacidad contratada en el proveedor lejano y por no usar toda la capacidad existente en el proveedor local.



PenalidadMenos	Tiempo											
Recurso	1	2	3	4	5	6	7	8	9	10	11	12
R1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
R2	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15

Tabla A4.15. Página de los sobrecostes de sub-utilización de las capacidades.

En la Tabla A4.16. se muestra la página de coste de *back order*. En esta página se coloca el coste de entregas diferidas. Sólo el producto A tiene entregas diferidas y sus costes aumentan a medida que se acerca el final de la temporada de ventas.



Coste BackOrder	Periodo											
SKU	1	2	3	4	5	6	7	8	9	10	11	12
A	0	0	0	0	0	0	0	0	0	1	2	3
B	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0

Tabla A4.16. Página de los costes de las entregas diferidas.

Para la experimentación del apartado 4.2.3. Efecto de la variación del plazo de entrega, los costes de fabricación del proveedor lejano varían en función del plazo de entrega del proveedor lejano (el plazo de entrega del proveedor local se mantiene constante e igual a un periodo, por ello su coste de fabricación también permanece constante). En la Tabla A4.17. se muestran los costes unitarios de fabricación.

Plazo de entrega	5	4	3	2
Proveedor lejano	10	11	13	14
Proveedor local	15	15	15	15

Tabla A4.17. Costes de fabricación de cada proveedor en función del plazo de entrega.

Los costes de fabricación incluyen los costes de producción y transporte. Los demás costes no cambian con el plazo de entrega del proveedor lejano.