UNIVERSIDAD DE SEVILLA

ESCUELA SUPERIOR DE INGENIEROS

SIMULACIÓN DE SISTEMAS DE PRODUCCIÓN MEDIANTE ARENA 3.0

ALEJANDRO JAVIER TOSINA GONZÁLEZ.

2001

Índice.

Objetivos y Alcance del Proyecto.

El presente proyecto tiene como principal objetivo realizar un acercamiento lo más detallado posible hacia el mundo de la Simulación y la utilidad que de ella se puede conseguir.

La Simulación es una herramienta cuya utilización se está extendiendo dentro de los entornos industriales para el análisis de sistemas complejos, fundamentalmente en entornos productivos y en empresas de servicios. Su enorme potencia permite la optimización de los recursos disponibles o el diseño de nuevos sistemas.

Pero no solo adquiere importancia en el mundo de la industria sino que los conceptos y métodos que esta herramienta proporciona disfrutan de gran aplicabilidad en otros muchos campos como las Telecomunicaciones, la Economía o incluso la Política.

Para afrontar este estudio se comienza con una breve pero detallada introducción al concepto de Simulación, a su utilidad y a los métodos y herramientas que emplea.

Posteriormente, y debido a la necesidad de implementar el proceso de Simulación sobre computadoras, se pretende realizar una presentación de los distintos lenguajes de programación que pueden utilizarse para realizar simulaciones de sistemas de eventos discretos, que son los que se van a estudiar, distinguiendo los que son de propósito general de aquellos que han sido desarrollados específicamente para realizar tareas de Simulación.

En concreto se va a estudiar con mayor detalle el lenguaje de simulación Arena 3.0, que es la herramienta empleada para la realización del presente proyecto.

Como aplicación donde exponer todas las potencialidades de la Simulación se analizará un sistema de producción, en concreto un Sistema de Fabricación Flexible (FMS), donde la informática y la robótica adquieren un papel relevante a la hora de optimizar dicho sistema. Será esta optimización la que se buscará a la hora de utilizar las técnicas de Simulación.

Por último es interesante destacar la gran utilidad y amplio uso que de esta herramienta se hace en la realidad para comprender la importancia y el interés que tiene este proyecto.

Introducción a la Simulación.

¿Qué es la Simulación?

El término *Simulación* se puede definir como la imitación del funcionamiento de un proceso o sistema real. Se trata pues, de diseñar un modelo matemático o lógico de ese sistema y experimentar sobre dicho modelo para describir, explicar y predecir el comportamiento del sistema real.

Existen situaciones en las que el sistema que va a ser modelado es suficientemente simple como para que podamos estudiar su comportamiento mediante el uso de métodos matemáticos. La solución de estos métodos consistirá en una serie de parámetros numéricos que nos darán una medida de las prestaciones del sistema.

No obstante, muchos de los sistemas que se presentan en el mundo real son tan complejos que resulta prácticamente imposible estudiarlos desde un punto de vista puramente matemático. En estos casos resulta necesario realizar simulaciones numéricas o basadas en el uso del ordenador para poder imitar el comportamiento de estos sistemas.

La Simulación no sólo permite estudiar sistemas reales para ver su comportamiento y, si procede, mejorarlos, sino que también permite comprobar el funcionamiento de sistemas que todavía no existen para, de esta forma, crearlos de la mejor manera posible.

En definitiva, la simulación consiste en imitar el comportamiento de sistemas reales, normalmente a través del uso del ordenador. De hecho, este aprovechamiento que hace de los ordenadores, cada vez más avanzados, motiva que la simulación sea una herramienta tan potente y tan utilizada en múltiples campos de la industria y la investigación.

¿Cuándo usar la Simulación?

La disponibilidad de lenguajes específicos de simulación, ordenadores de elevada velocidad de cálculo, con una disminución del coste de operación, y los avances en métodos de modelado y simulación han permitido que ésta sea una de las herramientas más ampliamente utilizadas tanto en la investigación de operaciones como en el análisis de sistemas.

Las técnicas de simulación se pueden utilizar por los siguientes motivos:

- La simulación permite el estudio y la experimentación de las interacciones internas dentro de un sistema complejo.
- Se pueden observar y estudiar cambios producidos en el sistema. Estos cambios pueden ser del entorno, organizativos o de información.
- El conocimiento adquirido en el diseño del modelo que se va a simular permite sugerir mejoras en el sistema que se está investigando.
- Cambiando las entradas al modelo durante la simulación se puede ver cómo va evolucionando su respuesta, así como estudiar qué variables son más importantes y cómo interactúan.
- La simulación permite comprobar soluciones obtenidas por procedimientos analíticos.
- Se puede experimentar con nuevos diseños y políticas de uso antes de la implementación en la realidad y, así, preparar para lo que pueda pasar.

De esta forma, la simulación se convierte en una herramienta especialmente útil en los siguientes campos:

- Operaciones de producción, mantenimiento y distribución en el entorno industrial.
- Análisis de tráfico en redes de carreteras.
- Estudio de flujos de personas en oficinas, aeropuertos, ...
- Simulación de la economía y predicción de decisiones políticas.
- Análisis de redes de datos o redes telefónicas.

Ventajas y desventajas de la Simulación.

Aunque la simulación es una herramienta adecuada para el análisis en muchos casos, es preciso no obstante considerar previamente las ventajas y desventajas que nos reporta.

Las principales **ventajas** de la simulación son las siguientes:

- Una vez que el modelo está construido, se puede utilizar repetidamente para analizar cambios en el diseño o diversas políticas.
- Suele ser menos costoso obtener datos de un proceso de simulación que de un sistema real.
- Los métodos de simulación son más fáciles de aplicar que los métodos analíticos.
- Los modelos analíticos normalmente requieren asumir muchas simplificaciones para hacerlos matemáticamente tratables. Los modelos de simulación no tienen estas restricciones.
- El entorno en el que se va a incluir el sistema puede ser controlado por el usuario.
- Si se produce algún fallo en el diseño del sistema es más barato corregirlo en un modelo simulado que en un sistema real.

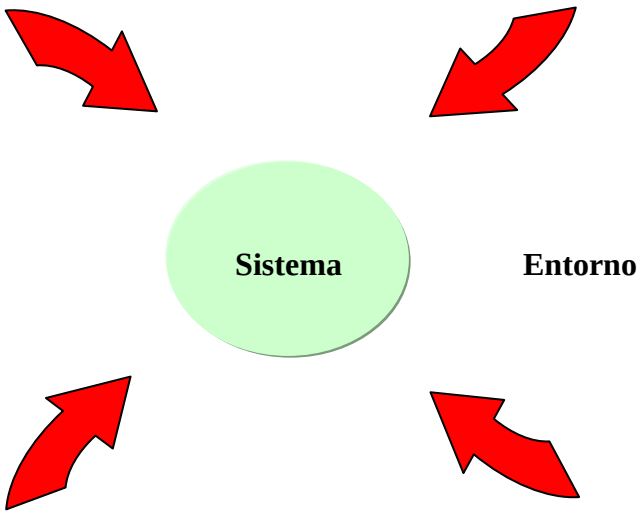
No obstante, la simulación también presenta una serie de **inconvenientes**:

- Es una técnica imprecisa, por ser aproximada. Se estudia un modelo, no el sistema real.
- Aumentar la precisión implica aumentar la complejidad del modelo. Esto conlleva un mayor coste de diseño.
- Pueden ser necesarias numerosas ejecuciones de la simulación, lo que implica un mayor coste de operación.
- Los usuarios que están acostumbrados a utilizar la simulación pueden despreciar el uso de técnicas analíticas aun en situaciones en las que estas últimas son apropiadas y suficientes.

Sistemas y su entorno de operación.

Un *sistema* se define como un grupo de objetos que se encuentran relacionados por algún tipo de interacción o interdependencia con el fin de cumplir un propósito determinado. Un ejemplo de esto sería una cadena de montaje dentro de una fábrica. Cada máquina es una entidad independiente de las demás pero relacionada con ellas para conseguir el objetivo de obtener el producto final.

Un sistema a menudo está afectado por cambios que se producen fuera de él. Estos cambios se denominan *entorno del sistema*. A la hora de modelar el sistema es necesario definir los límites entre dicho sistema y su entorno. Esta decisión puede depender del propósito del estudio que se vaya a realizar.



Dentro de un sistema es conveniente definir un conjunto de términos. Estos términos son los siguientes:

- *Entidad*: Es un objeto de interés dentro del sistema. Son componentes del sistema que se mueven, cambian de estado, afectan y son afectados por otras entidades. Son objetos dinámicos que se crean, se mueven y abandonan el sistema (temporales) o no (permanentes).
- *Atributo*: Es una propiedad de una entidad. En general, todas las entidades tienen los mismos atributos pero diferentes valores para diferentes entidades.
- *Recurso*: Son elementos del sistema por los que compiten las entidades. Cada entidad ocupa un recurso, lo usa y, cuando ha terminado, lo libera.
- *Cola*: Es el lugar donde esperan las entidades cuando no pueden moverse o cuando el recurso al que aspiran está ocupado.
- *Actividad*: Representa un periodo de tiempo en el que se realiza una función determinada.
- *Estado del sistema*: Se define como el conjunto de variables necesario para describir el sistema, dependiendo de los objetivos del estudio en cuestión. Cada variable es única para todo el modelo y no van asociadas a las entidades.
- *Evento*: Se define como un suceso instantáneo que puede cambiar el estado del sistema.

Tanto las actividades como los eventos pueden ser *endógenos*, si ocurren dentro del sistema, o *exógenos*, si se producen en el entorno del sistema pero le afectan de alguna forma.

Los sistemas, desde el punto de vista de su estado, pueden ser de dos tipos:

- *Discretos*: En estos sistemas las variables de estado cambian únicamente en instantes discretos de tiempo.
- *Continuos*: Las variables de estado cambian de forma continua a lo largo del tiempo.

Una vez definidos todos estos términos y conceptos podemos establecer que un sistema real está formado por un conjunto de componentes, relacionados a través de una estructura y afectados por un entorno determinado.

Diversos ejemplos de sistemas son los siguientes:

- Fábricas, cadenas de montaje.
- Servicios públicos, bancos, hospitales, oficinas,...
- Logística y distribución.
- Sistemas de ordenadores y de telecomunicación.
- Operaciones militares.
- Obras de infraestructura.
- Economía de una región.
- Etc...

Será objetivo de la simulación medir la calidad de estos sistemas, estudiar su comportamiento y, si no existe, diseñarlo de forma que funcione de la mejor manera posible.

Modelo de un Sistema.

Para estudiar un sistema, a veces es posible experimentar con el sistema en cuestión. No obstante, esto no siempre es posible. El nuevo sistema puede no existir todavía o puede ser demasiado caro o difícil experimentar con él. Puede ser incluso imposible o no deseable. Por este motivo es normal estudiar los sistemas a partir de un modelo obtenido a partir de él.

Un *modelo* se define como la representación de un sistema realizada con el objetivo de estudiarlo. Por este motivo no es necesario tener en cuenta todos los detalles del sistema que representa sino solo aquellos que realmente son necesarios para el estudio al que se va a someter. De esta forma, un modelo no solo es una representación de un sistema sino que también es una simplificación de éste. No obstante este modelo debe ser suficientemente detallado como para permitir la obtención de conclusiones válidas sobre el sistema real.

Para desarrollar un modelo correcto de un sistema es necesario antes entender perfectamente cómo funciona dicho sistema realmente. Este proceso es bastante útil a la hora de proponer cambios y posibles mejoras, además de permitir realizar un modelo bastante exacto del sistema.

Los modelos están compuestos, al igual que los sistemas, de entidades, atributos, actividades,... pero sólo de aquéllas que sean significativas para el estudio que se va a realizar.

Los modelos pueden ser de varios tipos:

- *Físicos* o *icónicos*: Cuando el sistema y el modelo se parecen físicamente. Un ejemplo serían las maquetas, tanto en miniatura como a escala real o los simuladores de vuelo.
- *Lógicos* o *matemáticos*: Son conjuntos de aproximaciones y suposiciones, tanto estructurales como numéricas, sobre la forma en la que trabaja el sistema que se modela. Suelen representarse en algún tipo de software.

Una segunda clasificación los divide en los siguientes grupos:

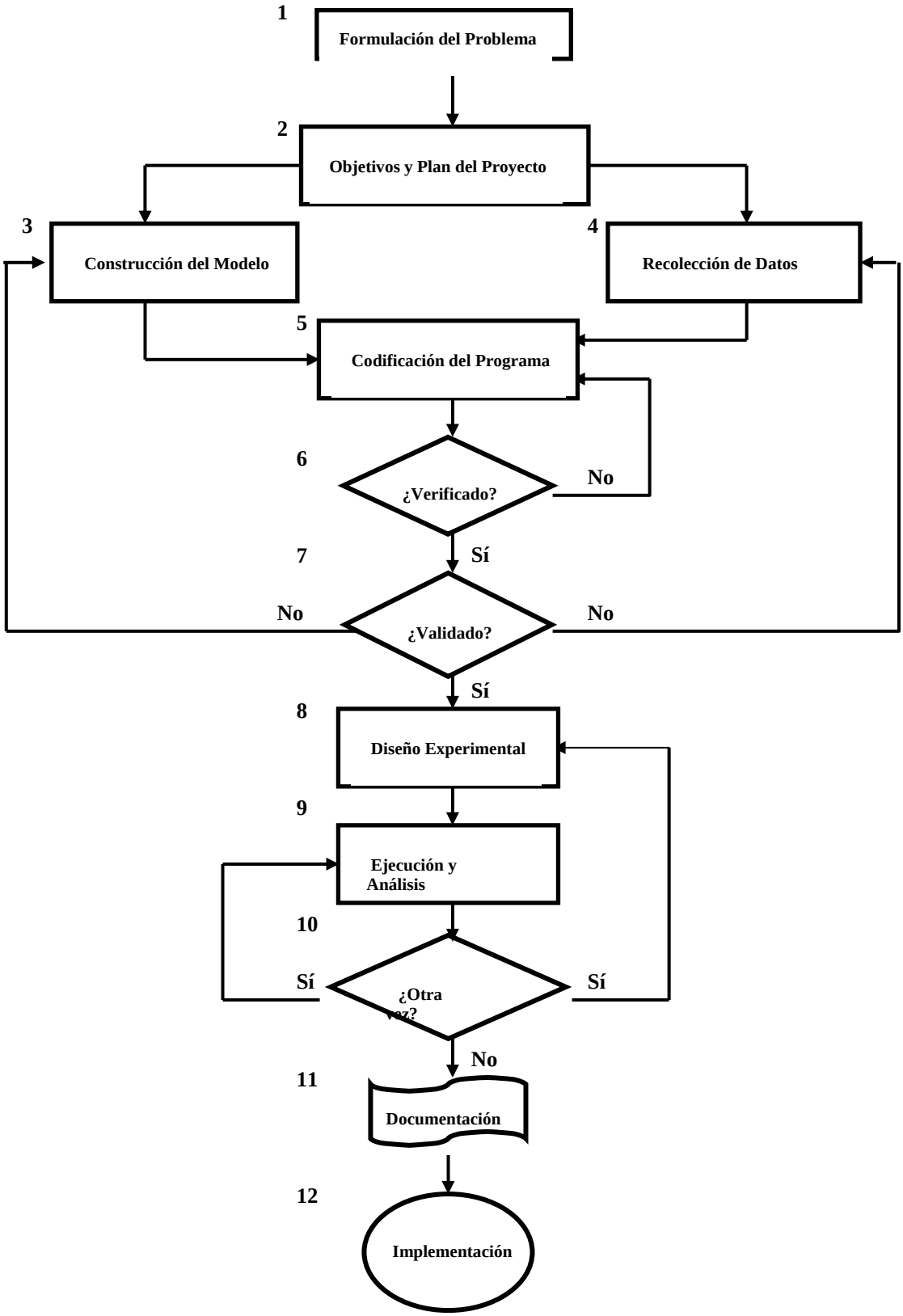
- *Dinámicos* o *Estáticos*: Si interviene el tiempo o no.
- *Continuos* o *Discretos*: Si el estado del sistema cambia en instantes determinados o de forma continua en el tiempo.
- *Estocásticos* o *Deterministas*: Si los valores son aleatorios o no los son.
- De *Ciclo Abierto* o de *Ciclo Cerrado*: Si las entidades entran y salen del sistema o si permanecen en el mismo.

El tipo de modelo que se elige para representar un sistema determinado depende de la propia naturaleza del sistema en cuestión. Así, si en el sistema no hay variables aleatorias se clasificará como determinista. Esto podría pasar en una consulta con cita previa, por ejemplo. Si, por el contrario, existe alguna variable del sistema que tenga una cierta incertidumbre se tratará de un modelo estocástico, como podría ser una ventanilla de oficina.

Cabe resaltar que en el caso de modelos continuos o discretos la diferencia es más sutil puesto que hay sistemas que se pueden modelar de cualquiera de las dos formas o, incluso, de una forma mixta. Esto dependerá de la naturaleza del sistema y de los objetivos que se quieran alcanzar.

Etapas de un proyecto de simulación.

Los pasos que hay que realizar para desarrollar un proyecto de simulación pueden seguir el siguiente diagrama de flujo:

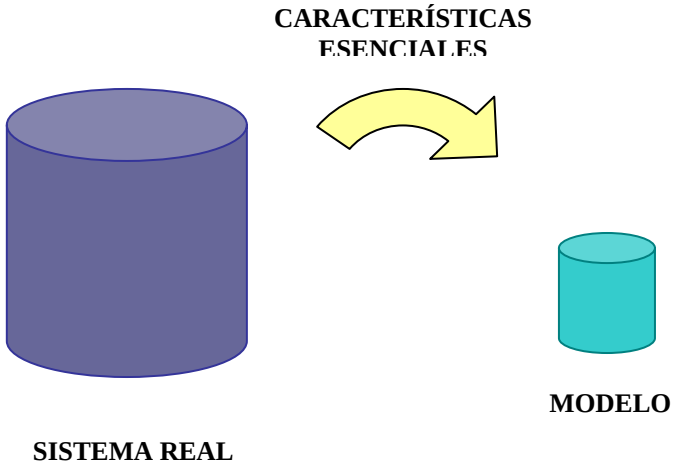


Una explicación más detallada de cada paso se da a continuación:

1.- *Formulación del problema*. Todo estudio debe comenzar con un establecimiento del problema. En este momento el analista debe comprender perfectamente todos los aspectos del sistema que va a estudiar. Hay que tener en cuenta que hay muchos casos en los que conviene reformular el problema a medida que se avanza en su estudio.

2.- *Establecimiento de los objetivos y plan del proyecto*. Los objetivos indican las cuestiones que se van a estudiar en la simulación. En este punto es donde se determina si la simulación va a ser una herramienta apropiada o no. Si se decide que sí es apropiada entonces sería recomendable tener en cuenta otras alternativas distintas que pudieran ser utilizadas como un método de evaluación o verificación. Para terminar el plan del proyecto es necesario tener en cuenta su coste, el número de personas involucradas en el mismo y el número de días que llevará realizar cada fase del proyecto.

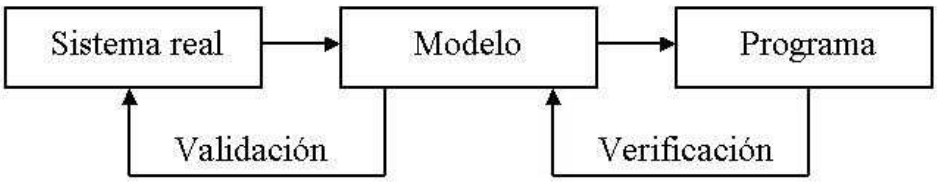
3.- *Construcción del modelo*. Para conseguir un buen modelo del sistema que se va a someter a estudio es preciso abstraer las características esenciales del problema, seleccionar y modificar de forma adecuada suposiciones básicas que caracterizan el sistema y, entonces elaborar el modelo para conseguir unos resultados que, aunque sean aproximados, resulten útiles.



4.- *Recopilación y preparación de los datos*. Este paso suele producirse simultáneamente con la propia construcción del modelo. No obstante, como la recopilación de los datos necesita un porcentaje bastante grande del tiempo total del proceso de simulación, es necesario comenzar cuanto antes. Los objetivos del estudio que se va a realizar dictaminan el tipo de datos que se va a necesitar.

5.- *Codificación del programa*. La mayoría de los sistemas reales son tan complejos que los modelos que se pueden hacer de ellos requieren una gran cantidad de información y una elevada capacidad de operación. Por este motivo el modelo que se desarrolle debe ser programado en un ordenador. El analista del sistema debe decidir si realiza el programa en un lenguaje de propósito general, FORTRAN, C, Pascal, Basic, o utiliza algún lenguaje específico de simulación como SIMAN o Arena.

6.- *Verificar el programa*. La verificación del programa consiste en comprobar que el programa está libre de errores y de acuerdo al modelo. Esta operación se puede realizar manualmente, gráficamente o mediante un test con soluciones ya conocidas.

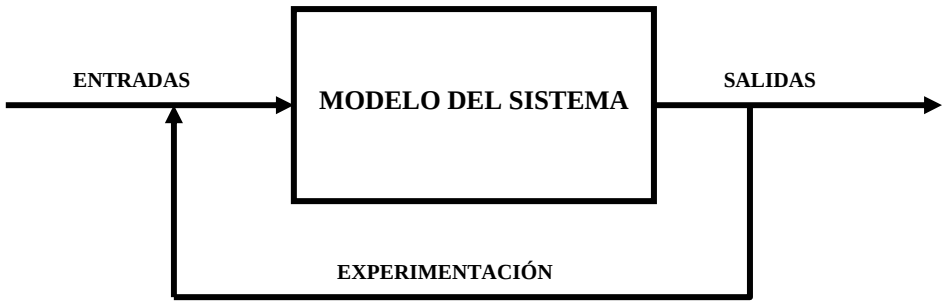


7.- *Validar el modelo.* Un modelo será válido si es una representación suficientemente aproximada del sistema. El modelo se puede validar con experimentos de campo o con una serie de métodos específicos como el *Test de Turing* o el *Método Delphi*.

8.- *Diseño experimental*. El diseño se puede realizar a dos niveles:

- Nivel estratégico: Se fijan los valores óptimos de los parámetros y éstos se relacionan con variables de salida.
- Nivel táctico: Se establecen las condiciones iniciales de régimen permanente y se fija el tiempo de ejecución y el número de ejecuciones.

9.- *Ejecución de la simulación y análisis de los resultados*. En este paso se determinan las medidas de las prestaciones del diseño del sistema que se está simulando. Se aportan una serie de entradas al modelo y se comprueba si lo que se obtiene es lo que realmente se desea o, por el contrario, hay que cambiar las entradas o el modelo mismo. Se produce así un proceso de realimentación.



10.- *¿Más simulaciones?* En función de los resultados de las simulaciones que ya se han completado, el analista decide si son necesarios más experimentos o si, por el contrario, las ya completadas son suficientes.

11.- *Documentación del programa y presentación de los resultados*. Habrá que documentar el programa por la posibilidad de que posteriormente vaya a ser utilizado por otros analistas, distintos de los que lo desarrollaron. La presentación de resultados se realizará en función del tipo de problema que se resuelve y de para quién vaya dirigida esta documentación:

- Problema de tipo no recurrente: El horizonte es de medio o largo plazo, las decisiones son únicas y los usuarios finales son de nivel medio-alto.
- Problema de tipo recurrente: Aquí se produce una presentación continua de resultados. Son decisiones rutinarias dirigidas a usuarios finales de tipo operador.

12.- *Implementación*. El éxito de la fase de implementación depende de lo bien que se realicen las fases anteriores del proyecto. Si el modelo y las suposiciones iniciales son correctas entonces la implementación se realizará con éxito.

Simulación de Eventos Discretos.

En este proyecto se centra el estudio en la Simulación de Eventos Discretos, mediante el modelado de sistemas en los cuales el estado de sus variables cambia solamente en instantes discretos de tiempo.

Para simular este tipo de sistemas se van a utilizar métodos numéricos más que analíticos. Los métodos analíticos son los que resuelven los modelos a partir del empleo de razonamientos matemáticos. Los métodos numéricos, por su parte, utilizan procedimientos informáticos para resolver los problemas matemáticos.

En el caso de simulación de modelos con métodos numéricos, estos modelos son ejecutados en vez de solucionados. Se trata de realizar una historia artificial del sistema a partir de las suposiciones del modelo para que, de esta forma, las observaciones obtenidas sirvan para comprobar el correcto funcionamiento de dicho sistema.

Se procurará que las operaciones individuales (llegadas, servicios,...) ocurran exactamente como en el sistema real. Los movimientos, interacciones y cambios sobre el sistema se producirán en el instante y orden programado. De esta forma se fuerza al modelo a actuar como la realidad.

Por último destacar que el trabajo más difícil que hay que realizar en Simulación es la obtención del modelo, lo que requiere un profundo conocimiento del sistema que se va a estudiar. Una vez modelado el sistema, el ordenador y el software de simulación realizará el resto.

Lenguajes de Simulación.

Introducción.

Para desarrollar un proyecto de simulación es preciso contar con un ordenador y un software suficientemente potente que permita desarrollar todas las posibilidades que ofrece esta herramienta de la simulación para estudiar sistemas que, en nuestro caso, serán de eventos discretos.

Existen varias formas distintas de trasladar el modelo realizado al ordenador. Éstas son las siguientes:

- Desarrollar un programa que podrá ser escrito en un lenguaje de propósito general como Pascal, C ó Java, o, de otra forma, en un lenguaje específico de simulación.
- Emplear un sistema de modelado interactivo visual (VIMS). Este sistema es una herramienta visual que permite desarrollar el modelo lógico seleccionando iconos de una plantilla utilizando el ratón. Ejemplos de esta forma de trabajar pueden ser MicroSaint y Witness.

La elección de un determinado software de simulación dependerá de una serie de factores que se pasan a analizar:

- El tipo de aplicación: Los distintos paquetes de software de simulación no suelen servir para todos los casos posibles ya que están orientados hacia un tipo de sistema específico. Esto no quiere decir que no se puedan utilizar para simular sistemas para los que no han sido específicamente diseñados pero dicha simulación requerirá de un mayor esfuerzo creativo por parte del analista que lo quiera utilizar.

- La finalidad que se pretende: No es lo mismo desarrollar un modelo que va a simular el propio analista que realizar un producto para un cliente que va a ejecutar la simulación pero que no sabe cómo se ha modelado el sistema o cómo funciona internamente el software que se ha utilizado. En el primer caso con obtener una serie de resultados bastará pero en el segundo habrá que disponer de una interfaz agradable al usuario y de una presentación de los datos clara y concisa.
- Conocimiento, política de software y soporte técnico al usuario.
- Precio.

El uso de lenguajes de simulación permite realizar aplicaciones de una forma más fácil, rápida y eficaz puesto que están orientados a facilitar este tipo de programación. No obstante, como contrapartida presentan una serie de desventajas puesto que son lenguajes que presentan una menor flexibilidad, un mayor coste y necesitan un tiempo de aprendizaje previo al desarrollo de la simulación.

Tipos de lenguajes de simulación de eventos discretos.

A continuación se va a realizar una clasificación de los tipos de lenguajes de simulación sobre la base del modo de empleo de cada tipo de software. Estos lenguajes se pueden clasificar en los siguientes grupos:

- Programados en lenguajes de propósito general.
- Programados en lenguajes de simulación.
- Sistemas estructurados en bloques.
- Sistemas de modelado interactivo visual (VIMS).

A excepción del VIMS, gran parte de estos lenguajes ha estado utilizándose durante más de 30 años de una forma u otra. Diferentes generaciones de productos ya existentes aparecen cada cierto tiempo aportando cambios que se producen sobre todo a nivel de programación.

Sistemas programados en lenguajes de propósito general.

En los primeros días de la simulación sobre ordenadores la única posibilidad consistía en realizar los modelos programando en algún lenguaje de propósito general como el FORTRAN, el C o el Pascal.

Hoy en día, a pesar de la existencia de lenguajes específicos de programación y de entornos visuales, se siguen utilizando estas técnicas por varios motivos:

- Coste del producto. Hay veces en las que una determinada organización o empresa no puede costear un software comercial de simulación.
- Coste de aprendizaje. Algunas personas preferirán programar en un lenguaje que ya conocen en vez de aprender uno nuevo.
- Se consigue mayor velocidad de ejecución y el programa puede interaccionar con otros que no tienen que ser específicamente de simulación (hojas de cálculo, presentaciones,...).
- Se pueden programar situaciones muy específicas para las que el software de simulación disponible no esté preparado.

De todas formas estos lenguajes de programación presentan una serie de inconvenientes como son el coste en tiempo y en esfuerzo de programación. Estos lenguajes no están adaptados a realizar este tipo de tareas de simulación por lo que la programación será más complicada.

Para facilitar el desarrollo de programas de simulación con este tipo de lenguajes es recomendable utilizar algún tipo de librerías que se encarguen de las tareas más comunes como el avance de tiempo, la gestión de secuencias de sucesos o el muestreo, y, de esta forma, hacer más cómoda la programación de estos sistemas de simulación.

Sistemas programados en lenguajes de simulación.

Un lenguaje debe estar preparado para indicar al ordenador lo que el programador quiere que haga. Los lenguajes orientados a un problema específico tienen una sintaxis que está bien ajustada a las tareas que vamos a solicitar. Como los lenguajes de propósito general no tenían esta sintaxis ha sido necesario el desarrollo de lenguajes específicos de simulación.

Las características de los lenguajes de simulación son las siguientes:

- Ficheros ejecutables que implementan las secuencias de eventos que discurren durante la simulación.
- Sintaxis bien ajustada al propósito de la simulación. Se trata de un conjunto de estructuras de datos que soportan las tareas que son comunes en simulación. De esta forma se permite programar sistemas con una menor cantidad de líneas de código, con menos errores potenciales, y en una menor cantidad de tiempo.
- Tratamiento de datos y seguimiento de variables. De esta forma se puede depurar el programa o presentar los resultados de la simulación al final de la ejecución de la misma.
- Apoyo a la experimentación. Normalmente estos lenguajes disponen de algún tipo de *shell* de control para realizar los experimentos sobre el modelo cambiando parámetros o presentando los datos de las diferentes simulaciones.

Dos ejemplos de lenguajes específicos de simulación son SIMSCRIPT o MODSIM.

SIMSCRIPT.

Este lenguaje de simulación comenzó en los años 60 como una forma de facilitar a los no especialistas el diseño de programas de simulación.

La idea original era que SIMSCRIPT se comportara como un preprocesador para FORTRAN. De esta forma el analista utilizaba la sintaxis de SIMSCRIPT para codificar el modelo y, luego, un traductor generaba el código FORTRAN. La sintaxis de SIMSCRIPT era más potente que la de FORTRAN y esto facilitaba las tareas de programar modelos y simularlos. Para ejecutar el programa de simulación el programa FORTRAN se compilaba y podía ser ejecutado desde el sistema SIMSCRIPT.

Posteriores versiones de SIMSCRIPT permiten modelado de sistemas basados en eventos y basados en procesos, en los cuales la lógica que sigue una aplicación básica consiste en bloques de procesos que definen una secuencia cronológica de actividades en las cuales se desarrollan entidades de distintas clases.

MODSIM.

MODSIM, cuya versión actual es MODSIM III, está concebido como un lenguaje orientado a objetos que encuentra su raíz en SIMULA. De esta forma este lenguaje de simulación presenta características comunes a todos lo lenguajes orientados a objetos como son las siguientes:

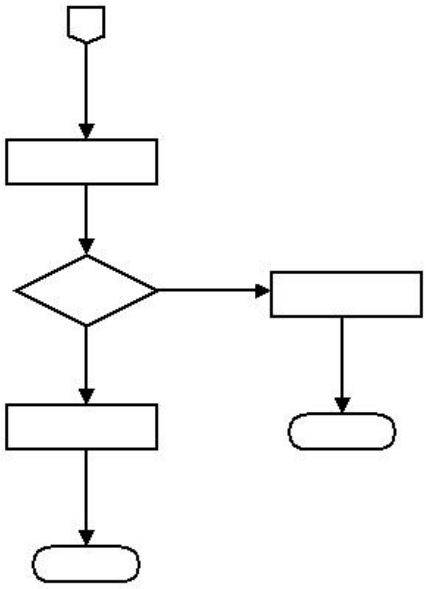
- Mecanismo de clases. Todas las variables son objetos que pertenecen a clases determinadas que están predefinidas o pueden ser creadas por el programador. Una clase consiste en un segmento de código que tiene dos partes, estructuras de datos y funciones de operación sobre esas estructuras de datos. La definición de una clase normalmente incluye funciones miembro, métodos, que se utilizan para cambiar el valor de las variables miembro o para observarlas.
- Entidades encapsuladas. Las definiciones y declaraciones de clase están contenidas en módulos de código por los que son relativamente independientes.
- Polimorfismo. Esta propiedad permite que determinadas funciones se puedan utilizar por objetos diferentes aportando parámetros de distinto tipo.

MODSIM aplica todas estas propiedades de los lenguajes orientados a objetos para realizar programas de simulación.

Sistemas estructurados en bloques.

Para desarrollar un programa de simulación utilizando algún lenguaje de propósito general o algún lenguaje de simulación es necesario que el analista que realiza el modelo sea un buen programador. Por este motivo aparecen los sistemas estructurados en bloques o de diagrama de flujo.

El objetivo del desarrollo de este tipo de sistemas era permitir que los no programadores pudieran desarrollar simulaciones de eventos discretos. Esto se consiguió definiendo una serie de elementos de flujo que interactuaban para realizar la simulación. Consiste en desarrollar un diagrama de flujo para representar el sistema que va a ser modelado.



Dos ejemplos característicos de este tipo de lenguajes son GPSS y SIMAN.

GPSS.

Originalmente se trataba de producir un sistema que pudiera simular redes de telecomunicaciones de forma que pudiera ser utilizado por ingenieros que no tuviesen una gran experiencia como programadores.

Un modelo de un sistema en GPSS consta de transacciones que fluyen sobre una red. Estas transacciones son equivalentes a entidades y los nodos de la red son equivalentes a puntos en los que el progreso de las entidades pueden ser retrasados en su ciclo de vida.

Aunque GPSS está clasificado como un sistema estructurado en bloques normalmente no se suelen dibujar los diagramas de flujo. En este sentido GPSS es considerado a veces como un lenguaje de simulación. En definitiva un programa de GPSS consta de una secuencia de comandos, cada uno de los cuales corresponde con un tipo de bloque distinto.

GPSS utiliza la siguiente terminología:

- Transacciones: Son entidades temporales del sistema que son creadas y pueden ser destruidas a lo largo del proceso de simulación. Estas transacciones se mueven a lo largo de un conjunto de bloques que definen el proceso de estas entidades temporales.
- Facilidades: Son entidades permanentes del sistema que pueden ser usadas para representar los recursos que necesitan las transacciones en los nodos de la red.

GPSS proporciona una forma rápida y bastante potente de desarrollar algunos tipos de modelos de simulación, especialmente aquellos basados en redes de colas en las cuales hay relativamente pocas interacciones entre clases de transacciones.

SIMAN

SIMAN es un lenguaje de simulación especialmente apropiado para modelar sistemas de eventos discretos, continuos o una combinación de ambos.

Este lenguaje, al igual que GPSS, formula los problemas a través de diagramas de bloques. De esta forma, un sistema desarrollado en SIMAN puede ser descrito como una interacción de procesos para lo cual se utilizan diagramas de bloques interconectados entre sí y en los que cada bloque representa una función. Así las entidades fluyen a través de los bloques que definen componentes del sistema.

A diferencia de GPSS, SIMAN reconoce y separa explícitamente las tres fases principales del proceso de simulación: definición del modelo, experimentación y análisis de resultados. Esta característica permite una gran versatilidad a la hora de realizar la experimentación sin provocar cambios significativos en el código del modelo.

Otra diferencia de SIMAN con respecto a otros lenguajes de simulación basados en diagramas de bloques consiste en la facilidad y relativa rapidez que permite a la hora de modelar sistemas mediante el uso de bloques especiales para propósitos concretos.

Un conjunto fundamental de bloques de SIMAN consiste en los siguientes:

- CREATE: Creación de una o varias entidades.
- QUEUE: Llegada de una entidad a una cola.
- SEIZE: Solicitud de uno o varios recursos por una entidad.
- DELAY: Retraso en el tiempo por la actividad de una entidad.
- RELEASE: Liberación de uno o varios recursos.
- TALLY: Registro de valores de una entidad para estadísticas.

Cada bloque en SIMAN tiene un símbolo que lo identifica dentro del diagrama de flujo. Las entidades que entran en el sistema irán recorriendo dicho diagrama de flujo pasando a través de los distintos bloques que lo componen. Además, es necesario un conjunto de elementos adicionales de experimentación que definen la ejecución concreta de la simulación. Estos elementos pueden ser:

- PROJECT: Definición del proyecto.
- DISCRETE: Número de elementos del proyecto.
- RESOURCES: Definición de los recursos.
- PARAMETERS: Definición de los parámetros.
- TALLIES: Definición de los registros de valores.
- REPLICATE: Número de ejecuciones y tiempo de ejecución.

La sintaxis de los elementos del modelo y de los elementos adicionales de experimentación se muestra a continuación junto con algunos ejemplos.

Elementos del modelo.

Etiqueta	Nombre Bloque, Operandos: Modificadores;	Comentarios
	CREATE,5:UN(3,2):MARK(2);	CUSTOMERS ARRIVE
	SEIZE:MACHINE;	CAPTURE MACHINE
	DELAY:EX(2,2)	PROCESS PART

Elementos de experimentación.

Nombre de elemento, Operandos:Modificadores;	Comentarios
PROJECT,JOBSHOP A, A. TOSINA, 23/11/00;	TITLE LINE
DISCRETE,5,2,3;	DEFINE MODEL SIZE
RESOURCES:1,MACHINE;	MACHINE CAPACITY = 1

Sistemas de modelado visual interactivo (VIMS).

Los sistemas VIMS se fundamentan en la existencia de sistemas operativos que presentan una interfaz gráfica de usuario (GUI) como Microsoft Windows entre otros.

Los modelos son creados utilizando el ratón para seleccionar objetos de simulación predefinidos en algún menú y colocarlos en la pantalla en el lugar apropiado. Los caminos que recorren las partes son creados en la pantalla dibujando líneas que unen los iconos de los servidores en una red lógica.

En gran parte del software de este tipo el usuario puede pulsar en el icono de algún elemento y desplegar una ventana con formularios donde se puede parametrizar el objeto en cuestión.

Muchos de estos lenguajes VIMS utilizan una red como modelo genérico sobre el que montar los desarrollos que se vayan realizando. De esta forma, las entidades van fluyendo a través de la red de nodo en nodo. En estos nodos su flujo sufre un retraso como consecuencia de alguna actividad que se realice en él en función de los recursos con los que cuente.

Ejemplos típicos de este tipo de sistemas podrían ser Witness o Arena. Este último entorno permite que SIMAN, que se definió anteriormente como un lenguaje de bloques, pueda ser tratado como VIMS, con todas las ventajas que ofrece el modelado visual.

El Lenguaje de Simulación Arena.

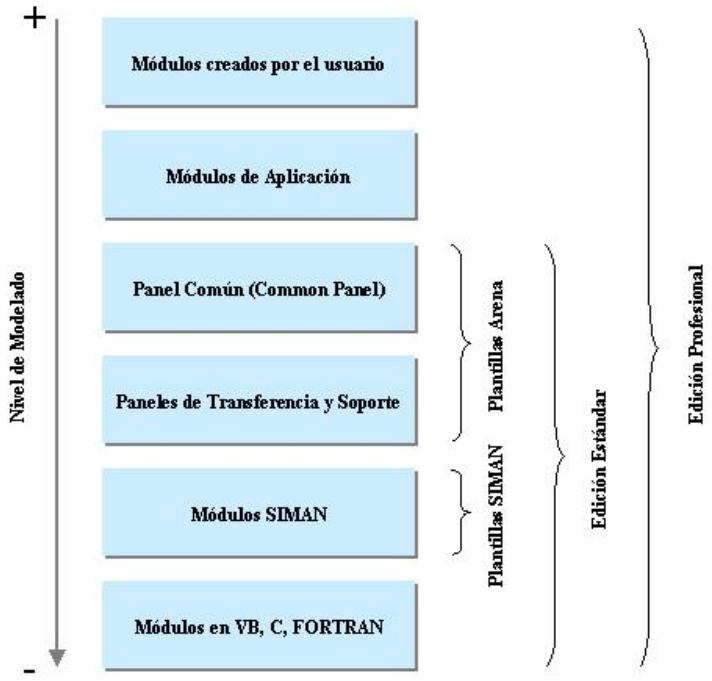
Introducción.

El lenguaje de simulación Arena, al igual que las otras herramientas VIMS, es un instrumento de simulación de "alto nivel", es decir, opera a través de interfaces gráficos muy intuitivos, como menús y diálogos, que pueden ser fácilmente utilizados mediante el uso del ratón. Los modelos se construyen partiendo de bloques constructivos básicos que se conectan entre sí. Posteriormente se ejecuta la simulación y se puede observar su evolución mediante un sistema de animación gráfica, comprobando como va cambiando el estado del sistema en el transcurso del tiempo de simulación.

Arena combina la facilidad de uso de herramientas VIMS con la flexibilidad de los lenguajes de simulación e incluso de los lenguajes de propósito general como Visual Basic, FORTRAN o C. Esto lo consigue proporcionando una serie de módulos de simulación gráfica y modelado intercambiables que se pueden combinar para obtener una amplia variedad de modelos de simulación. Para conseguir una mayor facilidad de uso de esta herramienta estos módulos están organizados en paneles que se agrupan formando plantillas. Mezclando plantillas distintas se puede acceder a una gran cantidad de módulos de construcción de modelos con enormes posibilidades.

Los módulos de Arena están compuestos de componentes SIMAN. De hecho, utilizando la edición profesional de Arena se pueden crear nuevos módulos para sistemas particulares. Estos módulos SIMAN se pueden utilizar conjuntamente con los módulos de alto nivel que tiene Arena e incluso con algunos desarrollados en lenguajes de alto nivel. Esta es una de las posibilidades que permiten que Arena sea una herramienta de gran versatilidad y potencia.

Los módulos que componen el conjunto de elementos que Arena permite utilizar para desarrollar modelos se estructuran de una forma jerárquica de la siguiente forma:



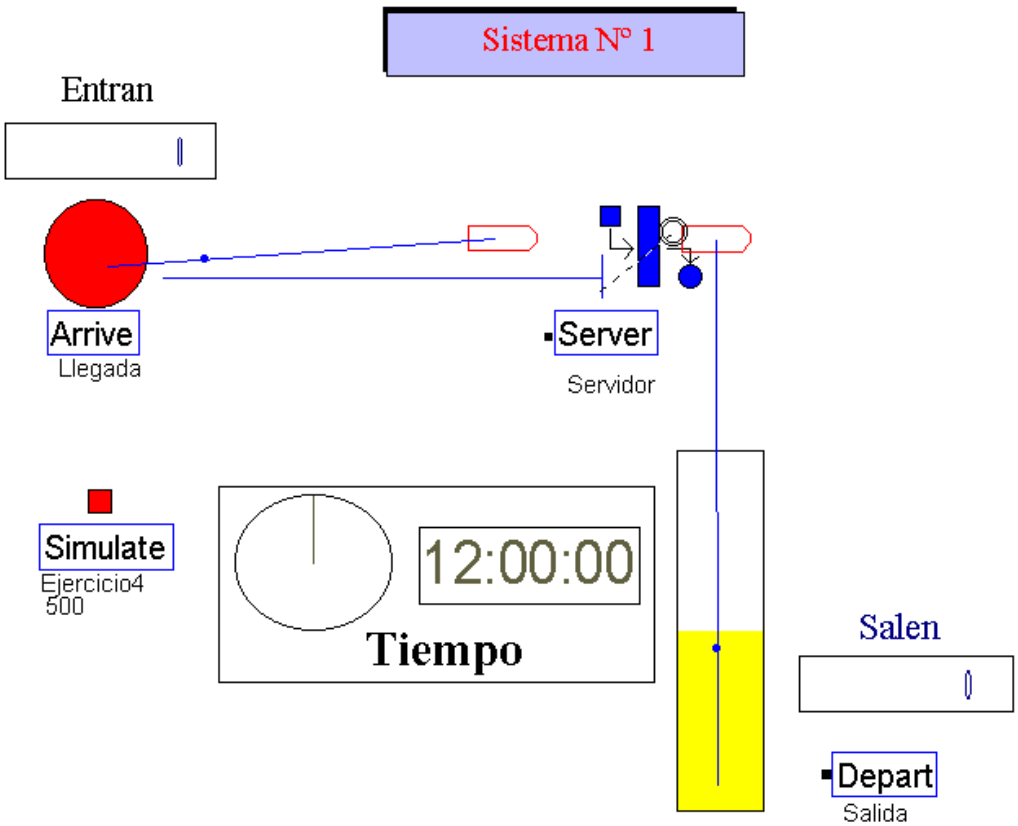
Como complemento a sus capacidades Arena permite a su vez la importación de ficheros de dibujo tipo DXF (AUTOCAD) como parte de su potente entorno gráfico para, de esta forma, observar el desarrollo de las simulaciones de una forma bastante agradable e intuitiva. También permite obtener una salida de datos de la simulación hacia una hoja de EXCEL para un posterior tratamiento de los datos y analizadores propios de datos de entrada y salida a fin de poder tratar con mayor profundidad los resultados de la simulación y, de esta forma, sacarle un mayor provecho al trabajo realizado.

Con todo lo comentado anteriormente Arena resulta ser un entorno de trabajo bastante versátil y potente a la hora de desarrollar trabajos de modelado y simulación. A continuación se expone un pequeño manual en el que se explica, a partir de una serie de ejemplos prácticos de Teoría de Colas y Secuenciación, el funcionamiento básico de esta herramienta y de sus módulos principales, para concluir el proyecto con el modelo de un sistema de producción complejo que permita observar todas las capacidades de esta herramienta.

Caso 1: Modelo de Cola y Servidor.

El sistema que se va a estudiar es un sistema con una cola y un servidor, cuya política de colas es FIFO (primero en llegar, primero en servir), al que acceden usuarios según una ley exponencial o de Poissón de tasa λ usuarios por minuto y cuya tasa de servicio es de μ unidades por minuto. El objetivo de este ejemplo consiste en comparar el caso en el que el tiempo de servicio sea de tipo exponencial de media 1/μ con el caso determinista donde es constante 1/μ.

El modelo del presente sistema realizado con Arena es el siguiente:



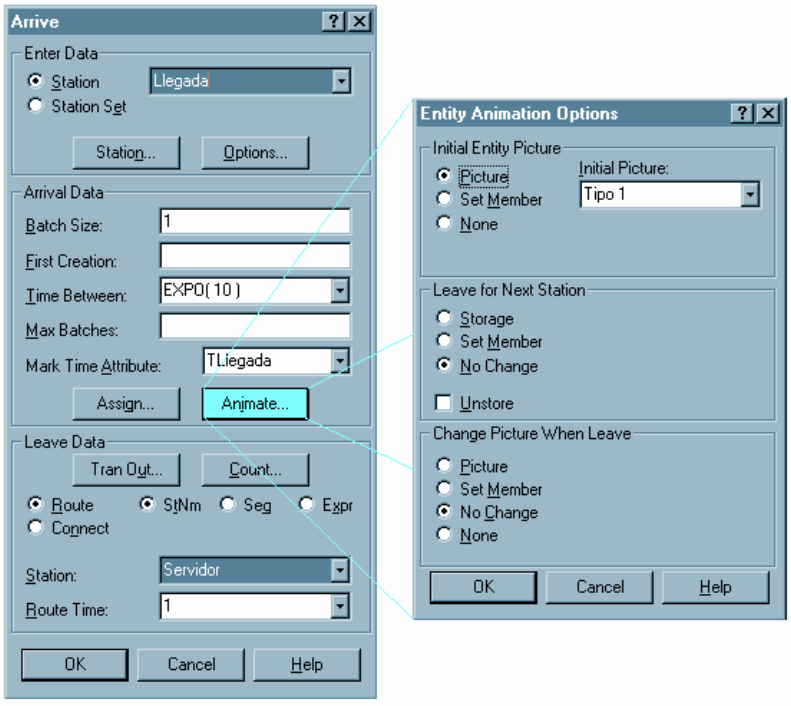
Subsistema de Llegadas.

El modelo consta de una población, representada por un círculo rojo, cuya tasa de llegadas es exponencial y de media $1/\lambda = 10$ minutos entre llegadas. Los usuarios de esta población se encaminarán al servidor de este sistema y serán atendidos según el tiempo de servicio que soliciten.

Para tener medidas sobre el tiempo de permanencia global en el sistema se va a imponer que se marque un atributo de tiempo en el instante en que lleguen los individuos al sistema, este atributo será *TLlegada*.

Para representar a las entidades de esta población en la simulación que posteriormente se realizará del sistema, se le asociará a cada una un icono de librería gráfica. Esto se indica en el submenú *Animate* del módulo *Arrive*.

El menú de selección de estos parámetros se muestra a continuación.



Para dirigir a los usuarios de la población desde la llegada hacia el servidor se va a definir un transporte de duración 1 minuto mediante *Route*, de forma que se pueda observar con claridad el flujo de entidades entre ambos elementos del modelo.

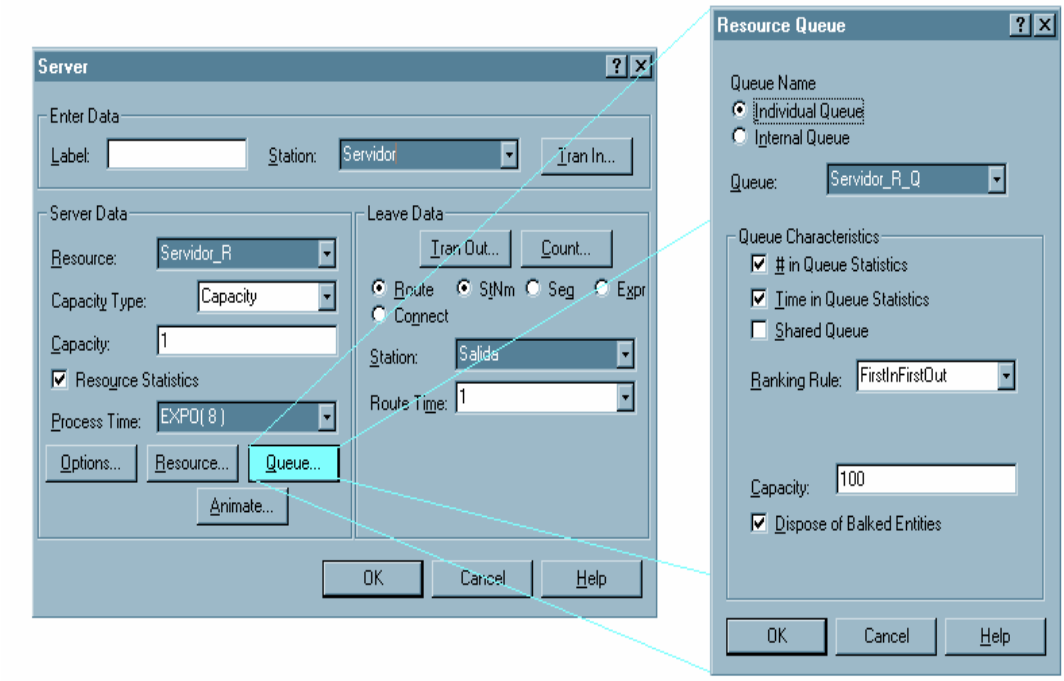
Subsistema Servidor.

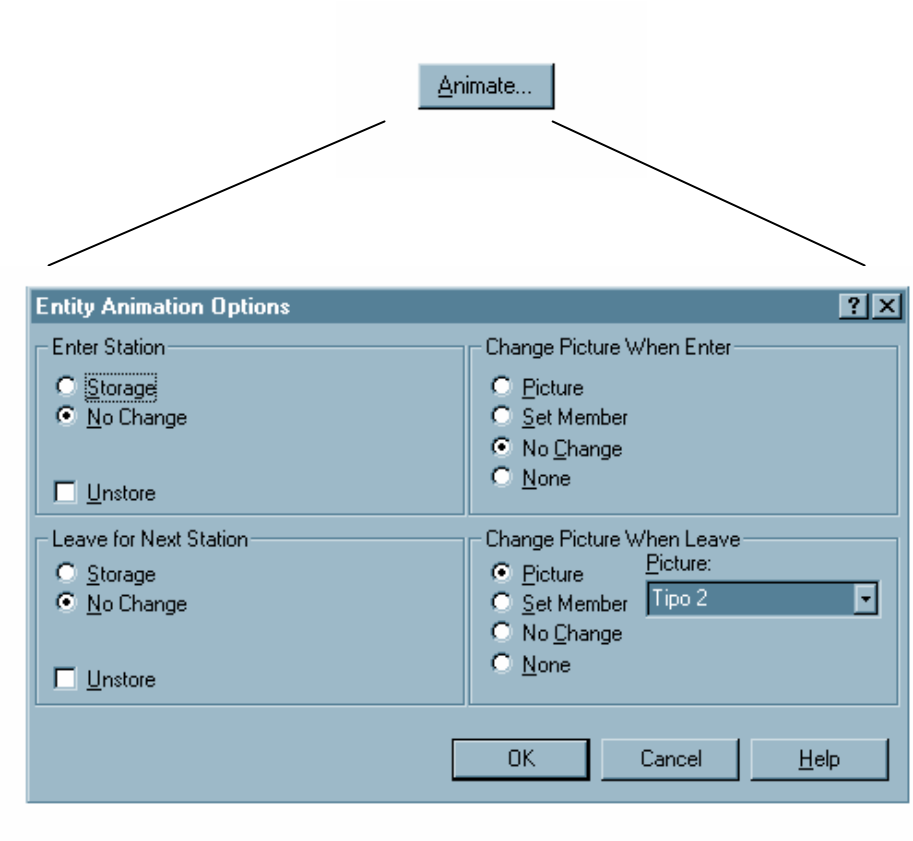
Este sistema consta de un único servidor monoestación, con cola de gran capacidad, al que irán llegando los individuos procedentes de la población de entrada y que solicitarán un tiempo de proceso que en un principio supondremos exponencial.

La disciplina de cola en este modelo será FIFO, es decir, los primeros en llegar serán los primeros en ser servidos.

Este servidor consta de un recurso *Servidor_R* de tipo *Capacity* de tamaño 1 (monoestación) al que se accede por una cola *Servidor_R_Q* que se ha definido de tamaño 100 y, como ya se ha dicho antes, disciplina FIFO.

En el bloque “Animate” se ha establecido que los usuarios de la población cambien de color (amarillo) una vez que hayan terminado de ser atendidas por el servidor.



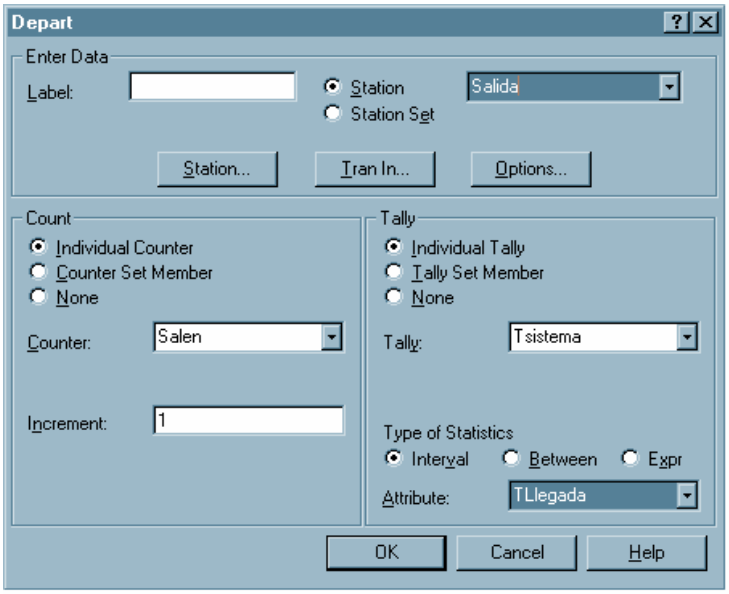

Animate...
Entity Animation Options
Enter Station
Storage
No Change
Unstore
Change Picture When Enter
Picture
Set Member
No Change
None
Leave for Next Station
Storage
No Change
Unstore
Change Picture When Leave
Picture
Set Member
No Change
None
Picture:
Tipo 2
OK
Cancel
Help

Subsistema de Salida.

Este subsistema consta de un módulo *Depart* que funciona como un sumidero de entidades.

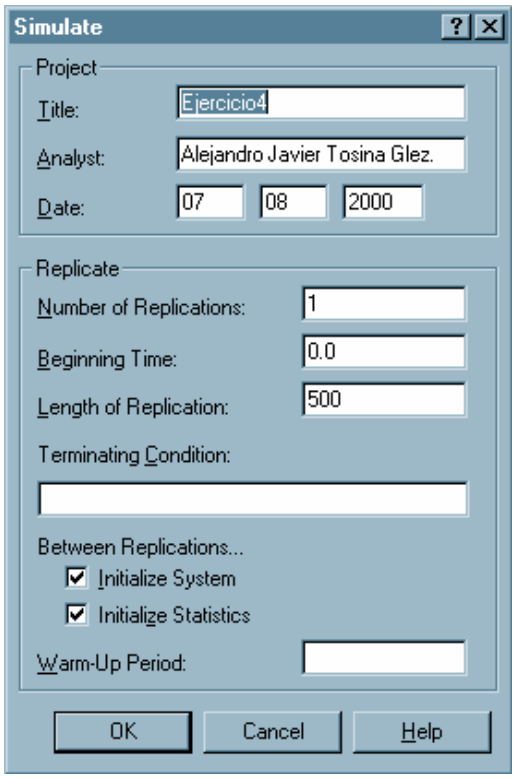
Lo único que se define aquí es un contador de entidades (Salen) que terminan de ser procesadas y salen del sistema y un *Tally* (Tsistema). Un *Tally* proporciona estadísticas del sistema. En este caso concreto se le ha indicado que presente el tiempo total de permanencia en el sistema de los individuos de la población. Para ello se marca la casilla *Interval* y en el atributo se indica el que se marcó al principio en el subsistema de llegada, *TLlegada*. De esta forma el *Tally* dará el intervalo de tiempo que las entidades han durado en el sistema desde que llegan (TLlegada) hasta que salen por el *Depart*.

El contador será un *Individual Counter* que se incrementará de uno en uno conforme las entidades vayan terminando su tiempo de proceso y salgan del sistema.

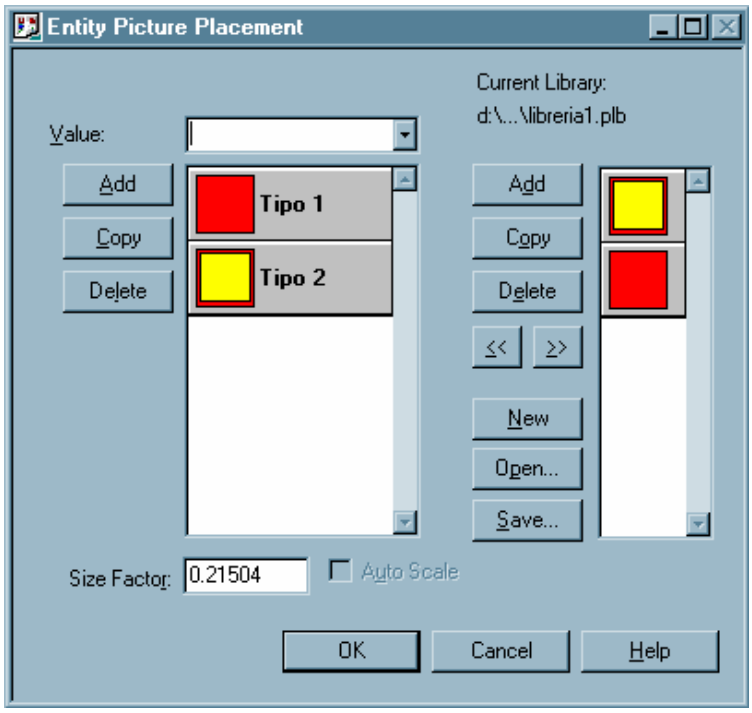


Subsistema de Simulación.

Este subsistema lo compone un solo bloque. Éste es un bloque *Simulation* en el que se tiene información útil para el desarrollo de la simulación del modelo como el número de veces que se va a simular el sistema, el momento de empezar y el intervalo de simulación que se va a considerar.



En este subsistema también se encuentra un icono que representa una de las figuras que se va a emplear en la animación gráfica de las entidades de la simulación. Si se pulsa dos veces con el ratón en este icono aparece un menú en el que se encuentran todas las figuras que se van a utilizar.



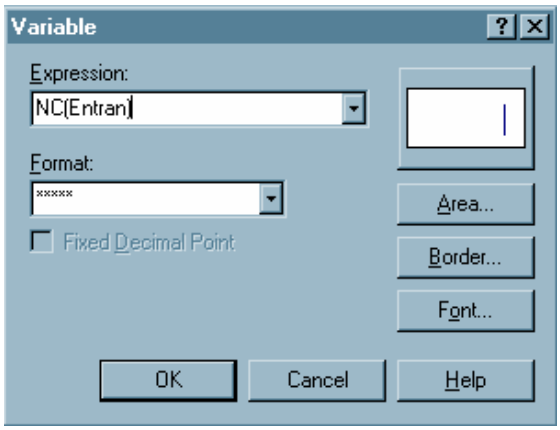
En esta aplicación, el cuadrado rojo (Tipo 1) representará a las entidades de la población que entran en el sistema. El cuadrado amarillo será de las entidades que terminan de ser atendidas por el servidor.

Estas figuras se asignan en el submenú *Animate* de los bloques *Arrive* y *Server* del modelo.

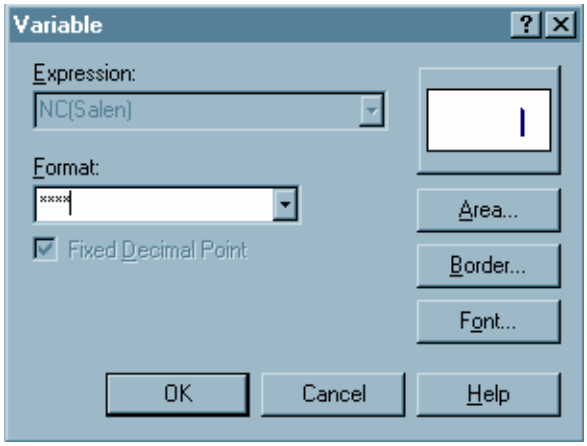
Otros elementos del modelo.

En este apartado se incluyen los contadores auxiliares que se han puesto en el modelo:

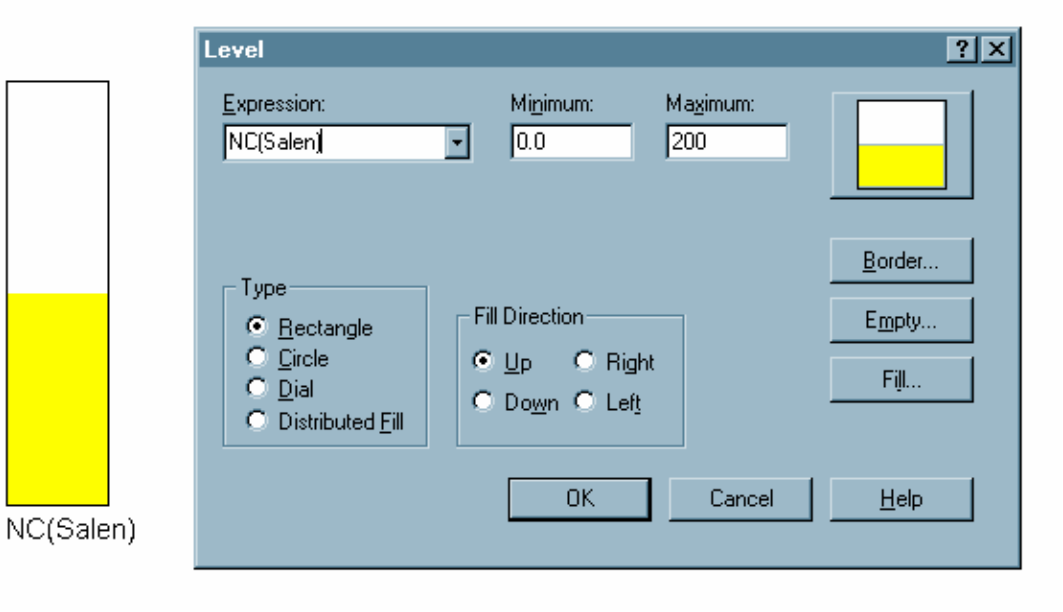
- Contador Entran: Se va incrementando con los usuarios que van apareciendo en nuestro sistema.



- Contador Salen: Número de entidades que terminan de ser atendidas por el servidor y salen por el módulo Salida.



En este subsistema auxiliar también se va a incluir una variable de nivel que simulará de forma gráfica el sumidero de entidades que es el módulo *Depart*. De hecho su expresión de comportamiento es NC(Salen) con lo cual se comporta de la misma forma que el contador asociado a las salidas de entidades del sistema.



Por último, en este modelo se incluye un reloj que da el tiempo que va durando la simulación. Proporciona el tiempo de simulación, no el tiempo real, que dependerá de la máquina en la que se esté ejecutando la simulación.

Simulación del Sistema.

Se procede a simular este sistema en el primer caso, aquél en el que el tiempo de servicio es exponencial y de media 8 minutos.

```
                    ARENA Simulation Results
              Alejandro Javier Tosina Glez. - License #9400000

                       Summary for Replication 1 of 1

Project:  Ejercicio4                         Run execution date :   9/12/2000
Analyst:  Alejandro Javier                   Model revision date:   7/ 8/2000

Replication ended at time      : 500.0

                                TALLY VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Observations
-------------------------------------------------------------------------------

Servidor_R_Q Queue Tim  46.868     (Insuf)    .00000     99.322         51
Tsistema                57.823     (Insuf)    3.1711     105.01         51

                           DISCRETE-CHANGE VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Final Value
-------------------------------------------------------------------------------

# in Servidor_R_Q       4.7805     (Insuf)    .00000     13.000     .00000
Servidor_R Available    1.0000     (Insuf)    1.0000     1.0000     1.0000
Servidor_R Busy         .91347     (Insuf)    .00000     1.0000     .00000

                                COUNTERS

                    Identifier                Count   Limit
                    ------------------------------------------

                    Salen                        51  Infinite
                    Entran                       51  Infinite


Simulation run time: 0.48 minutes.
Simulation run complete.
```

Se puede ver que el servidor está bastante cargado (ρ=0.91347) y que los tiempos de espera en cola y total en el sistema son bastante elevados (46.868 y 57.823 minutos respectivamente).

A continuación se verá que pasaría si el tiempo de servicio que solicitan es igual al tiempo medio entre llegadas. Si fueran valores constantes es de suponer que el servidor estaría ocupado el 100% del tiempo y que las entidades no tendrían que esperar nada en la cola (situación ideal). En este caso, al ser tiempos aleatorios, la cola crecerá hasta el máximo posible al no poder el servidor atender a tantos usuarios como lo solicitan. Una vez que la cola alcance su límite, si no es infinita, empezará a rechazar usuarios.

Los tiempos serán de 10 minutos.

Vemos la situación ideal.

ARENA Simulation Results
Alejandro Javier Tosina Glez. - License #9400000

Summary for Replication 1 of 1

Project: Sistema 1 — Run execution date : 9/12/2000
Analyst: Alejandro Javier — Model revision date: 7/ 8/2000

Replication ended at time : 500.0

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
Servidor_R_Q Queue Tim	.00000	(Insuf)	.00000	.00000	50
Tsistema	12.000	(Insuf)	12.000	12.000	49

DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
# in Servidor_R_Q	.00000	(Insuf)	.00000	1.0000	.00000
Servidor_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Servidor_R Busy	.99800	(Insuf)	.00000	1.0000	1.0000

COUNTERS

Identifier	Count	Limit
Salen	49	Infinite
Entran	51	Infinite

Simulation run time: 0.53 minutes.
Simulation run complete.

Se puede apreciar que, efectivamente, el servidor está ocupado aproximadamente el 100% del tiempo de simulación. No llega a este porcentaje exactamente debido al tiempo inicial que está esperando a que llegue la primera entidad.

El tiempo de permanencia en cola es nulo. Cuando una entidad llega al servidor, la anterior acaba de ser servida justo en ese momento.

No obstante, se observa que el tiempo total en el sistema no es de 10 minutos, como se estableció en el servidor. Esto sucede porque hay que considerar los dos minutos que tardan las entidades en recorrer los dos *Routes* que se han puesto entre las estaciones.

Se verá a continuación qué pasaría si, en vez de ser valores constantes, fuesen variables aleatorias tanto el tiempo entre llegadas como el tiempo de servicio. Se definirán como EXPO(10).

ARENA Simulation Results
Alejandro Javier Tosina Glez. - License #9400000

Summary for Replication 1 of 1

Project: Sistema 1 — Run execution date : 9/12/2000
Analyst: Alejandro Javier — Model revision date: 7/ 8/2000

Replication ended at time : 5000.0

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
Servidor_R_Q Queue Tim	119.36	(Corr)	.00000	295.20	473
Tsistema	130.95	(Corr)	2.0527	317.92	472

DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
# in Servidor_R_Q	12.578	(Corr)	.00000	39.000	39.000
Servidor_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Servidor_R Busy	.94672	.04841	.00000	1.0000	1.0000

COUNTERS

Identifier	Count	Limit
Salen	472	Infinite
Entran	512	Infinite

```
Simulation run time: 4.78 minutes.
Simulation run complete.
```

El tiempo de simulación ha sido aumentado 10 veces para ver el permanente. Se puede apreciar que, aunque el servidor está menos cargado que en el caso determinista, los tiempos de espera en cola (119.36 minutos) y de permanencia total en el sistema (130.95 minutos) son enormes y representan un servicio que se podría calificar de malo.

Nótese que al terminar el tiempo de simulación quedan en la cola 40 usuarios que todavía no han sido atendidos. Se puede ver que es un sistema bastante congestionado.

Se estudiará a continuación qué se debería hacer en cada caso para que la carga del servidor sea $\rho=0.5$.

Es de suponer que esto se conseguirá en el caso determinista haciendo el tiempo de servicio la mitad que el tiempo entre llegadas.

ARENA Simulation Results
Alejandro Javier Tosina Glez. - License #9400000

Summary for Replication 1 of 1

```
Project:  Sistema 1                          Run execution date :   9/12/2000
Analyst:  Alejandro Javier                   Model revision date:   7/ 8/2000

Replication ended at time      : 500.0
```

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
Servidor_R_Q Queue Tim	.00000	(Insuf)	.00000	.00000	50
Tsistema	7.0000	(Insuf)	7.0000	7.0000	50

```
                          DISCRETE-CHANGE VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Final Value
-------------------------------------------------------------------------------
# in Servidor_R_Q       .00000     (Insuf)    .00000     .00000     .00000
Servidor_R Available    1.0000     (Insuf)    1.0000     1.0000     1.0000
Servidor_R Busy         .50000     (Insuf)    .00000     1.0000     .00000

                                  COUNTERS

                    Identifier                Count   Limit
                    ----------------------------------------
                    Salen                        50  Infinite
                    Entran                       51  Infinite

Simulation run time: 0.18 minutes.
Simulation run complete.
```

Se comprueba que es correcta la suposición anterior. El servidor está cargado al 50% y los tiempos de espera en cola son nulos y en el sistema se tarda lo que en el servidor más los dos minutos que se pierden en los *Routes*.

Si se quiere buscar el mismo valor de carga en un sistema con tiempos aleatorios de distribución exponencial habrá que encontrar la tasa de llegadas que será necesaria para cumplir nuestro objetivo (suponiendo que el servicio solicitado no cambia y es EXPO(5) para todos).

Para unas llegadas EXPO(10):

```
                          ARENA Simulation Results
                    Alejandro Javier Tosina Glez. - License #9400000

                          Summary for Replication 1 of 1

Project:  Sistema 1                           Run execution date :   9/12/2000
Analyst:  Alejandro Javier                    Model revision date:   7/ 8/2000

Replication ended at time      : 500.0
                                   TALLY VARIABLES
Identifier              Average   Half Width  Minimum    Maximum   Observations
-------------------------------------------------------------------------------
Servidor_R_Q Queue Tim  5.3838     (Insuf)    .00000     36.654         48
Tsistema                12.401     (Insuf)    2.0442     40.126         47

                          DISCRETE-CHANGE VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Final Value
-------------------------------------------------------------------------------
# in Servidor_R_Q       .51932     (Insuf)    .00000     6.0000     1.0000
Servidor_R Available    1.0000     (Insuf)    1.0000     1.0000     1.0000
Servidor_R Busy         .46593     (Insuf)    .00000     1.0000     1.0000
```

```
                    COUNTERS

                Identifier                Count   Limit

                Salen                        47  Infinite
                Entran                       49  Infinite

Simulation run time: 0.27 minutes.
Simulation run complete.
```

Ahora el servidor está menos cargado que en el caso determinista, aunque tiene el inconveniente de que hay que esperar en la cola. Sin embargo los tiempos de espera son razonablemente bajos en este caso.

Se incrementará la velocidad de llegada al sistema para ver si se alcanza el valor de carga del 50%.

Llegadas EXPO(9):

```
                    ARENA Simulation Results
            Alejandro Javier Tosina Glez. - License #9400000

                    Summary for Replication 1 of 1

Project:  Sistema 1                          Run execution date :   9/12/2000
Analyst:  Alejandro Javier                   Model revision date:   7/ 8/2000

Replication ended at time      : 500.0


                              TALLY VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Observations
-------------------------------------------------------------------------------

Servidor_R_Q Queue Tim  3.9820     (Insuf)    .00000     21.850         52
Tsistema                11.050     (Insuf)    2.0263     30.424         52

                         DISCRETE-CHANGE VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Final Value
-------------------------------------------------------------------------------

# in Servidor_R_Q       .41414     (Insuf)    .00000     3.0000     .00000
Servidor_R Available    1.0000     (Insuf)    1.0000     1.0000     1.0000
Servidor_R Busy         .52709     (Insuf)    .00000     1.0000     .00000

                    COUNTERS

                Identifier                Count   Limit

                Salen                        52  Infinite
                Entran                       52  Infinite

Simulation run time: 0.03 minutes.
Simulation run complete.
```

La carga ya se acerca al valor deseado.

Llegadas EXPO(9.2):

ARENA Simulation Results
Alejandro Javier Tosina Glez. - License #9400000

Summary for Replication 1 of 1

Project: Sistema 1 Run execution date : 9/12/2000
Analyst: Alejandro Javier Model revision date: 7/ 8/2000

Replication ended at time : 500.0

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
Servidor_R_Q Queue Tim	3.6783	(Insuf)	.00000	21.825	50
Tsistema	10.729	(Insuf)	2.0263	30.191	50

DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
# in Servidor_R_Q	.36784	(Insuf)	.00000	3.0000	.00000
Servidor_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Servidor_R Busy	.50508	(Insuf)	.00000	1.0000	.00000

COUNTERS

Identifier	Count	Limit
Salen	50	Infinite
Entran	50	Infinite

Simulation run time: 0.03 minutes.
Simulation run complete.

Ahora se ha acercado de forma razonable al objetivo de ρ=0.5.

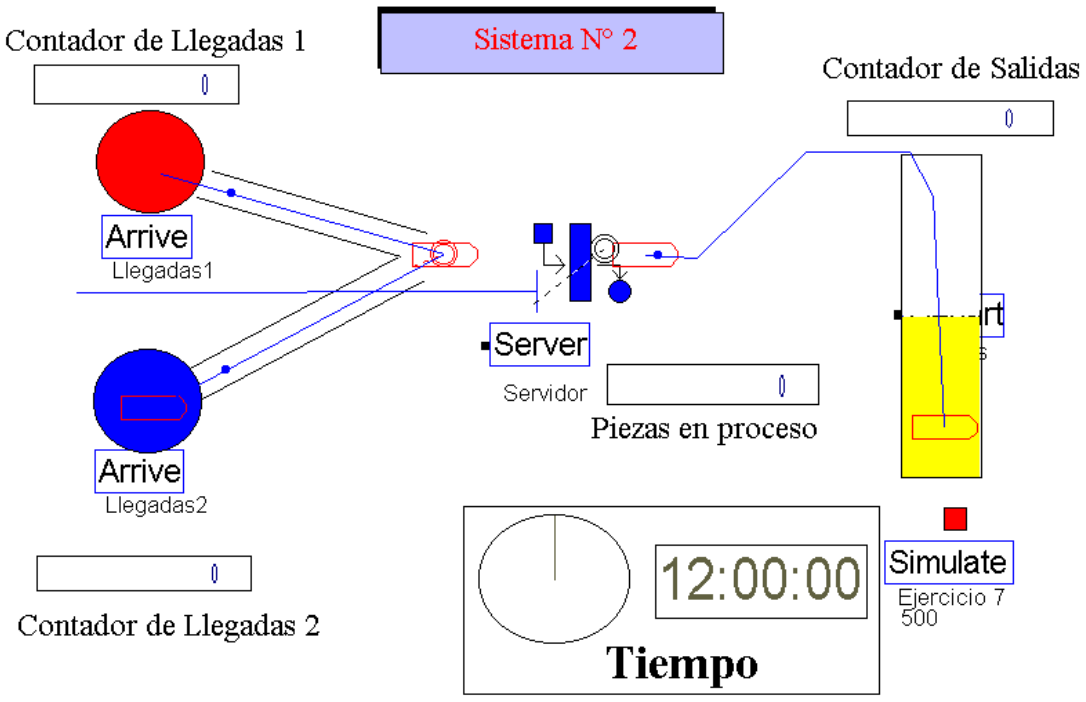
Después de analizar esta batería de simulaciones se llega a la conclusión de que para valores grandes de carga los tiempos de espera en el sistema son más pequeños en sistemas deterministas que en sistemas aleatorios. Sin embargo, cuando la carga se va reduciendo las prestaciones de los sistemas aleatorios se acercan cada vez más a los deterministas. En el límite de carga cero (o infinitésima) la relación entre ambos valores será de la unidad.

Caso 2: Clases de Población y Disciplinas de Cola.

Este sistema consta de una población que incluye dos tipos de usuarios que demandan servicios con una distribución exponencial de media 1/μ1 y 1/μ2 respectivamente.

El recurso es monoestación con cola indefinida. Las llegadas son poissonianas de tasas λ1 y λ2.

El modelo planteado es el siguiente.



Subsistema de Llegadas.

El modelo consta de un sistema con dos poblaciones, una que será representada con figuras de color rojo y otra de color azul, con tasa de llegadas exponencial de medias 5 minutos para la población roja y 10 minutos para la azul. Los individuos de estas poblaciones se encaminarán al único servidor del sistema, donde serán atendidos según el servicio que soliciten y la disciplina de cola que apliquemos.

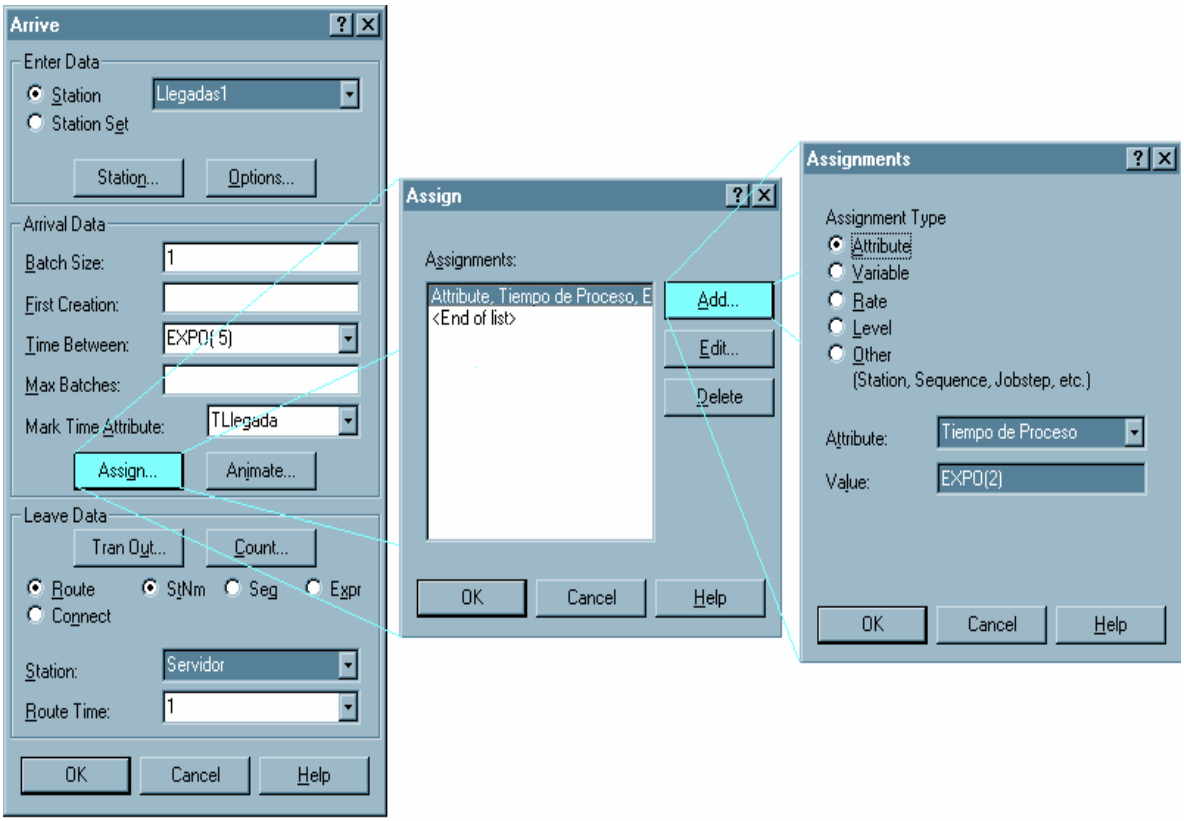
Para tener medidas sobre el tiempo de permanencia global en el sistema vamos a imponer que se marque un atributo de tiempo en el instante en que lleguen los individuos al sistema, este atributo será *TLlegada*.

El último aspecto importante que es necesario analizar para completar el subsistema de llegadas del presente modelo es la definición que se ha hecho de un atributo llamado *Tiempo de Proceso*, que indica el tiempo medio de servicio que solicita cada población. Este atributo incluirá una variable exponencial de media 2 minutos para la población roja y 7 minutos para la población azul. De esta forma, los usuarios de la población azul acapararán el sistema más tiempo que los usuarios de la población roja. Esto será importante cuando se tengan que definir disciplinas de cola.

La definición de todos estos parámetros se ha hecho activando algunos submenús del bloque *Arrive* correspondiente a cada población.

Con la Población 2, azul, sería lo mismo pero cambiando los valores de las distribuciones tanto de las llegadas como del atributo *Tiempo de Proceso*.

Entre el subsistema de llegadas y el servidor vamos a tener un *Route* que no es más que un camino entre ambos bloques, en el cual los usuarios de ambas poblaciones van a tardar en recorrerlo un tiempo de 1 minuto. Esto sólo va a servir para que durante la simulación del proceso se pueda ver claramente cómo evoluciona el flujo de usuarios entre los distintos bloques del modelo.


Arrive
Enter Data
Station
Llegadas1
Station Set
Station...
Options...
Arrival Data
Batch Size:
1
First Creation:
Time Between:
EXPO(5)
Max Batches:
Mark Time Attribute:
TLlegada
Assign...
Animate...
Leave Data
Tran Out...
Count...
Route
StNm
Seq
Expr
Connect
Station:
Servidor
Route Time:
1
OK
Cancel
Help
Assign
Assignments:
Attribute, Tiempo de Proceso, E
<End of list>
Add...
Edit...
Delete
OK
Cancel
Help
Assignments
Assignment Type
Attribute
Variable
Rate
Level
Other
(Station, Sequence, Jobstep, etc.)
Attribute:
Tiempo de Proceso
Value:
EXPO(2)
OK
Cancel
Help

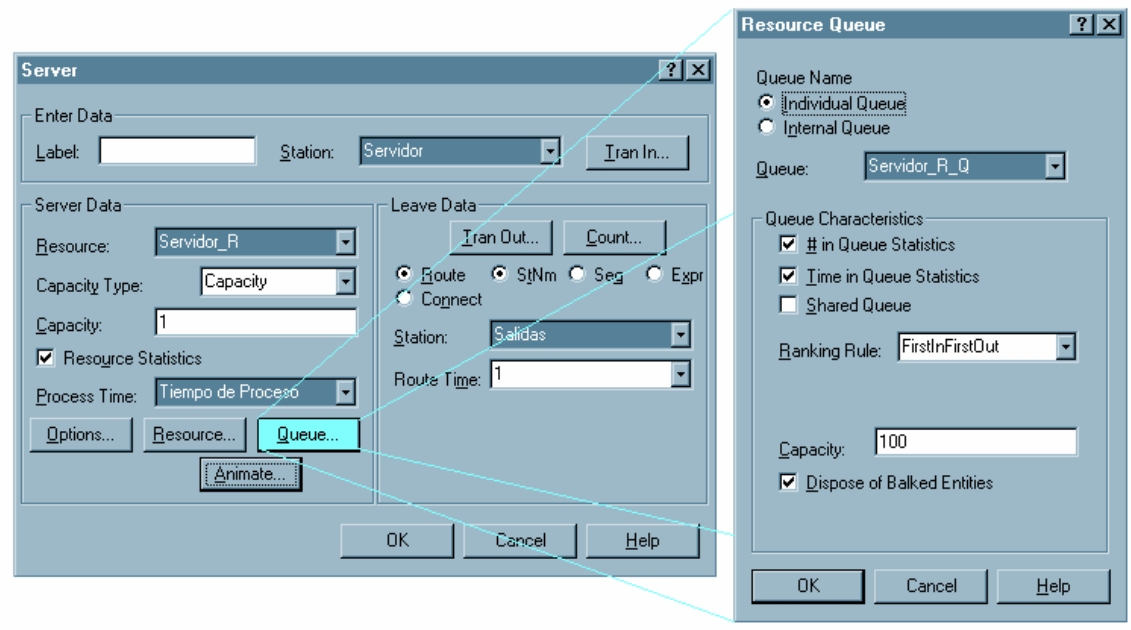
Subsistema Servidor.

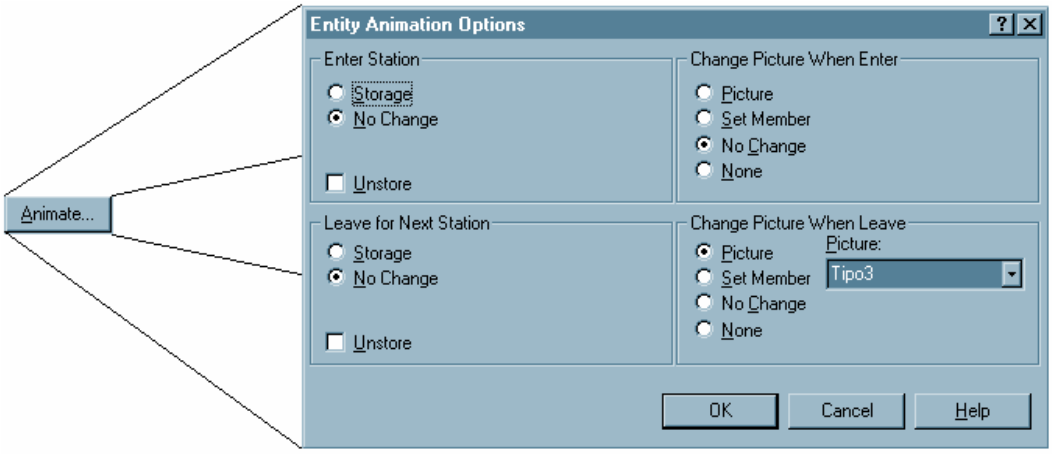
El presente sistema consta de un único servidor monoestación, con cola indefinida, al que irán llegando los individuos procedentes de las dos poblaciones y que solicitarán distinto tiempo de proceso, según de donde vengan.

Este servidor consta de un recurso *Servidor_R* de tipo *Capacity* de tamaño 1 (monoestación) al que se accede por una cola *Servidor_R_Q* que hemos definido de tamaño 100 y con disciplina FIFO, el primero en llegar será el primero en ser servido.

El tiempo que el servidor tarda en atender a los usuarios ha sido definido como el atributo *Tiempo de Proceso* explicado en el subsistema de llegadas.

En el bloque *Animate* se ha establecido que los usuarios de ambas poblaciones cambien de color (amarillo) una vez que han terminado de procesarse. Esto se ha hecho así para que se pueda ver gráficamente la comparación entre este modelo y un sistema real de procesamiento de elementos. Un producto entra en una máquina y sale algo distinto al haber sido procesado por dicha máquina.





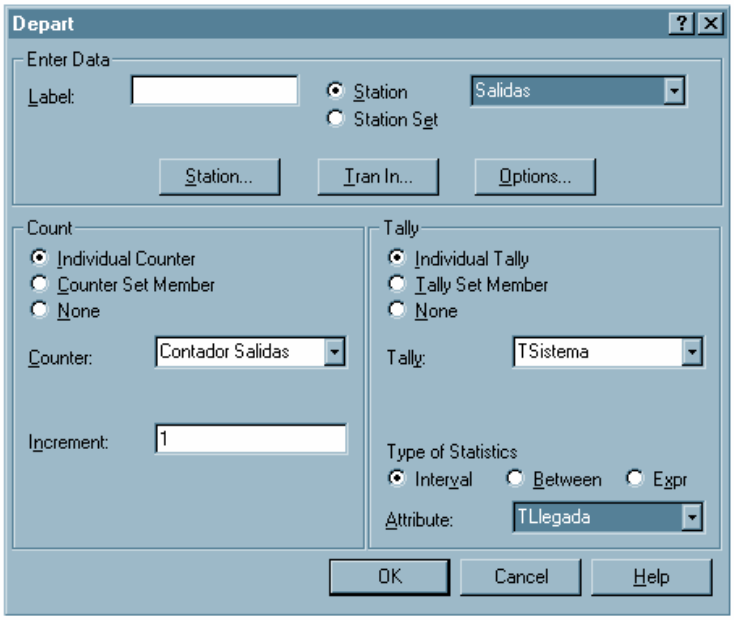
Una vez que los usuarios han sido atendidos pasan a un bloque de salida a través de un *Route* de duración 1 minuto.

Subsistema de Salida.

Este subsistema consta de un módulo *Depart* que funciona como un sumidero de entidades.

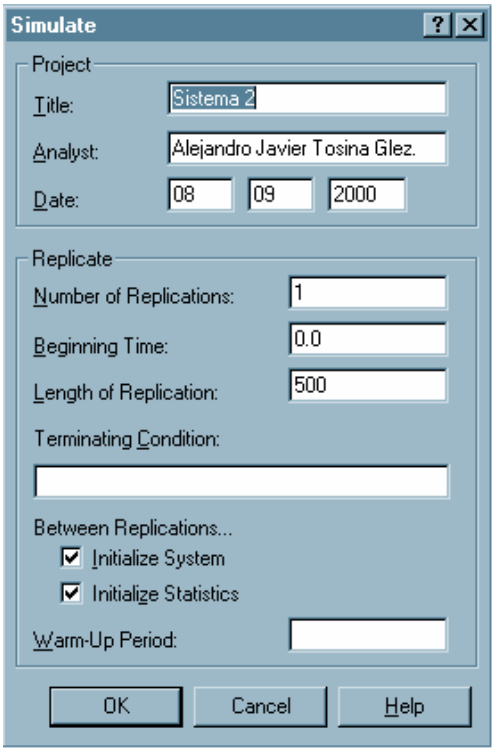
Lo único que se define aquí es un contador de entidades (Contador Salidas) que terminan de ser procesadas y salen del sistema y un *Tally* (Tsistema). Un *Tally* proporciona estadísticas del sistema. En este caso en concreto le hemos indicado que proporcione el tiempo total de permanencia en el sistema de los individuos de las poblaciones. Para ello se marca la casilla *Interval* y en el atributo se indica el que se marcó al principio en el subsistema de llegada, *TLlegada*. De esta forma el *Tally* dará el intervalo de tiempo que las entidades han durado en el sistema desde que activan *TLlegada* hasta que salen por el *Depart*.

El contador será un *Individual Counter* que se incrementará de uno en uno conforme las entidades vayan terminando su tiempo de proceso y salgan del sistema.

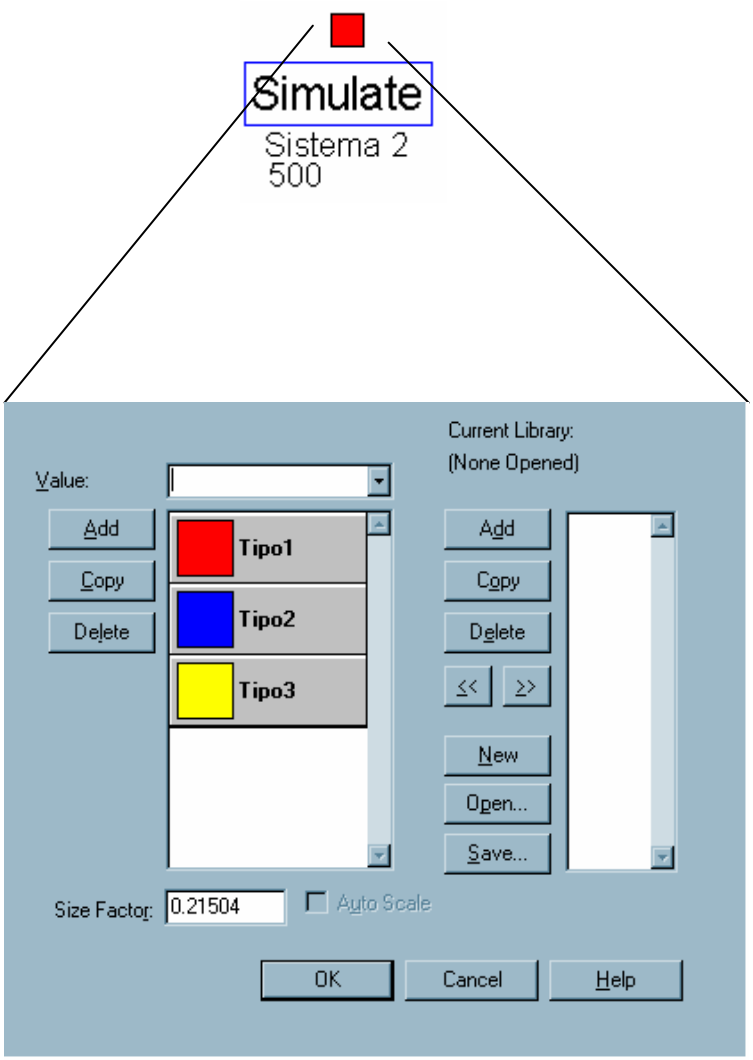


Subsistema de Simulación.

Este subsistema lo compone un solo bloque. Éste es un bloque *Simulation* en el que se tiene información útil para el desarrollo de la simulación del modelo, como el número de veces que se va a simular el sistema, el momento de empezar y el intervalo de simulación que vamos a considerar.



En este subsistema también se encuentra un icono que representa una de las figuras que se va a emplear en la animación gráfica de la simulación. Si se pincha dos veces con el ratón en este icono aparece un menú en el que se muestran todas las figuras que se van a utilizar.



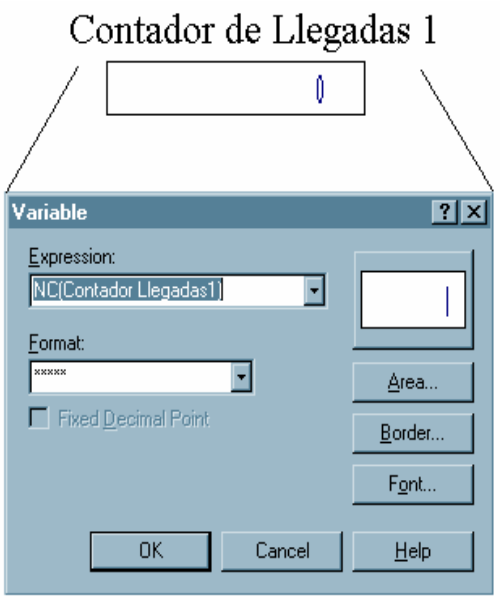
En la presente aplicación, el cuadrado rojo (Tipo 1) representará a las entidades de la población 1, el cuadrado azul (Tipo 2) será de la población 2 y el amarillo (Tipo 3) de las entidades que terminan de ser procesadas.

Estas figuras se definen en el submenú *Animate* de los bloques *Arrive* y *Server* del modelo.

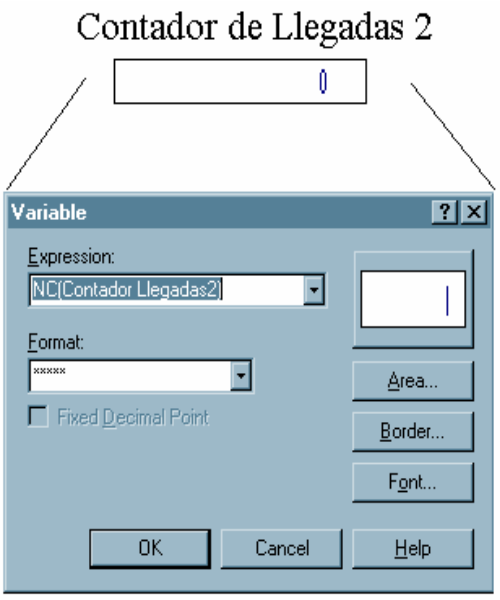
Otros elementos del modelo.

En ese apartado se incluyen los contadores auxiliares que se han utilizado. Estos son los siguientes:

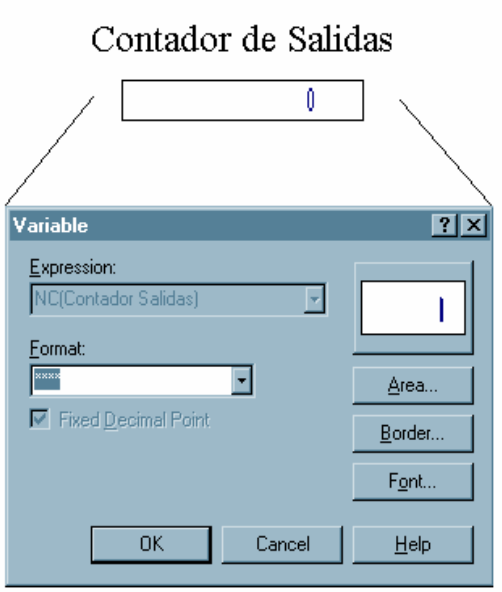
- Contador de Llegadas 1: Se va incrementando conforme van apareciendo entidades de la población 1 (roja).



- Contador de Llegadas 2: Número de entidades de la población 2 (azul) que van apareciendo.

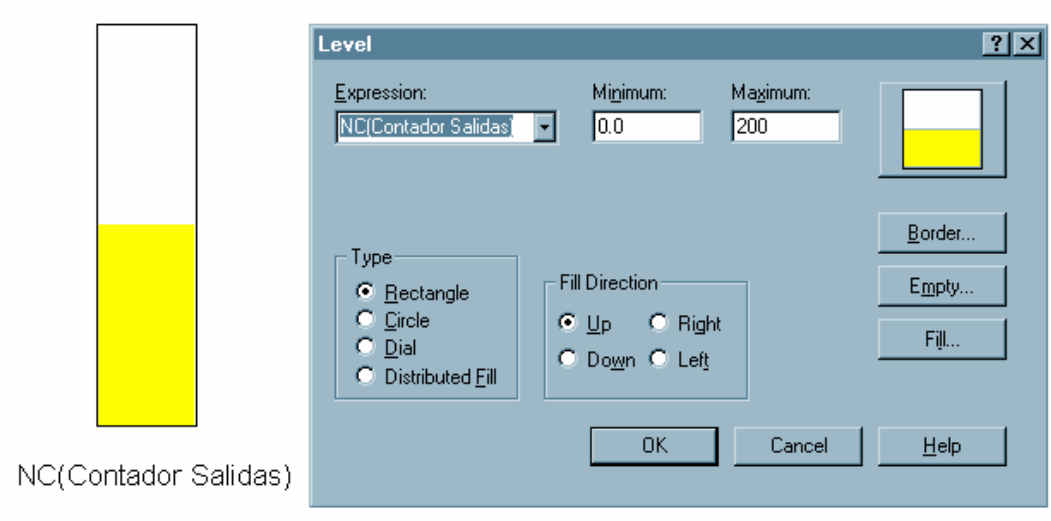


- Contador de Salidas: Ya definido anteriormente. Número de entidades que terminan de ser procesadas, de cualquier población, y que salen del sistema.



- Piezas en proceso: Indica el número de entidades que han entrado en el sistema pero que todavía no han salido. Esto incluye las que se encuentran en la cola de entrada al servidor, en medio del proceso, en el servidor e incluso las que han salido del servidor pero todavía no han llegado al módulo *Depart* ya que se tarda un minuto en recorrer el *Route* que enlaza el Servidor con la Salida.

En este subsistema auxiliar también se va a incluir una variable de nivel que simulará de forma gráfica el sumidero de entidades que es el módulo *Depart*. De hecho su expresión de comportamiento es NC(Contador Salidas) con lo cual se comporta de la misma forma que el contador asociado a las salidas de entidades del sistema.



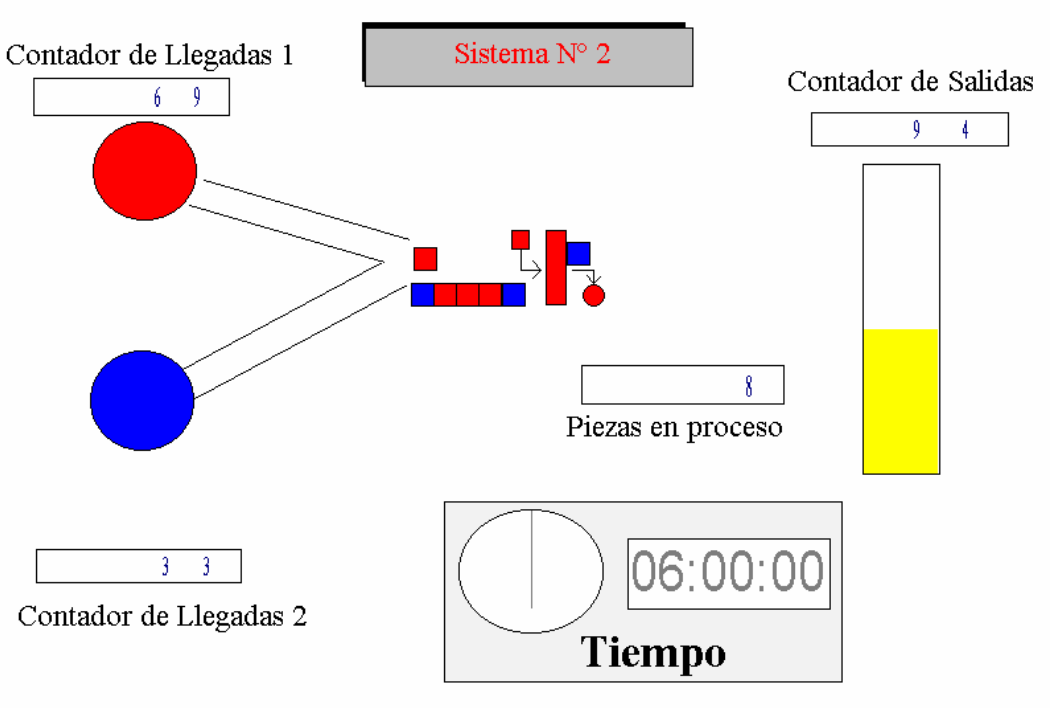
Por último, este modelo también incluye un reloj que nos da el tiempo que va durando la simulación.

Simulación del Sistema.

Una vez explicado el modelo utilizado se va a proceder a simularlo. Para ello habrá que tener en cuenta algunos aspectos:

- La disciplina de cola será en un principio FIFO (la que viene por defecto) con lo cual el servidor atenderá a las entidades según el orden en que vayan llegando.
- El tiempo de simulación se ha establecido en 500 minutos (algo más de 8 horas). Se está considerando tiempo de simulación. En tiempo real será bastante menos.

Se procede a ejecutar la simulación y los resultados son los siguientes:



```
                    ARENA Simulation Results
              Alejandro Javier Tosina Glez. - License #9400000

                    Summary for Replication 1 of 1

Project:  Sistema 2                          Run execution date :   9/10/2000
Analyst:  Alejandro Javier                   Model revision date:   8/ 9/2000

Replication ended at time      : 500.0
                                  TALLY VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Observations
-------------------------------------------------------------------------------

Servidor_R_Q Queue Tim  18.439     (Insuf)    .00000     64.492        126
TSistema                23.954     (Insuf)    2.1270     81.754        125

                        DISCRETE-CHANGE VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Final Value
-------------------------------------------------------------------------------

# in Servidor_R_Q       5.0454     (Insuf)    .00000     16.000     13.000
Servidor_R Available    1.0000     (Insuf)    1.0000     1.0000     1.0000
Servidor_R Busy         .92929     (Insuf)    .00000     1.0000     1.0000

                                  COUNTERS

                    Identifier                Count   Limit
                    ------------------------------------------

                    Contador Salidas            125  Infinite
                    Contador Llegadas1           91  Infinite
                    Contador Llegadas2           48  Infinite

Simulation run time: 0.78 minutes.
Simulation run complete.
```

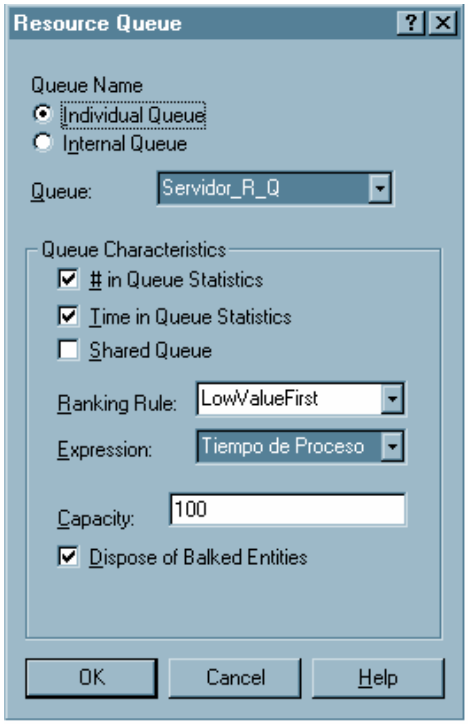
Como se puede comprobar, han entrado 139 entidades en el sistema, 91 de la población 1 (rojas) y 48 de la población 2 (azules). Esto es debido a la diferencia que se estableció entre las distribuciones de llegada de ambas poblaciones.

Del sistema salen 125 entidades. Las 14 que quedan son las que se encuentran en el sistema en el momento de parar la simulación.

El servidor ha estado bastante ocupado (ρ=0.92929) en este caso. El servicio que se le ha pedido en media ha sido de 5.0454 minutos.

El tiempo medio de permanencia en cola ha sido 18.439 minutos y el tiempo de permanencia en el sistema ha tenido un valor medio de 23.954 minutos.

Estos valores son bastante malos en cuanto a la calidad del servicio que se ofrece a las entidades, por lo que se verá qué pasaría si se utiliza una disciplina SEPT, es decir, que se atiende primero a los que piden menos, de esta forma es de suponer que los tiempos medios de permanencia en el sistema disminuirán sin que el servidor esté muy desocupado, lo que tampoco es deseable. Esto se consigue porque las entidades que piden más recursos al sistema no acaparan al servidor y las que piden menos pueden ser atendidas antes, con lo cual permanecen menos tiempo en el sistema.



Los resultados de la simulación son los siguientes:

```
                    ARENA Simulation Results
          Alejandro Javier Tosina Glez. - License #9400000

                  Summary for Replication 1 of 1

Project:  Ejercicio 7                       Run execution date :   9/10/2000
Analyst:  Alejandro Javier                  Model revision date:   8/ 9/2000

Replication ended at time      : 500.0

                              TALLY VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Observations
-------------------------------------------------------------------------------

Servidor_R_Q Queue Tim  6.7505     (Insuf)    .00000     114.70        133
TSistema                12.275     (Insuf)    2.1270     134.85        132
```

DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
# in Servidor_R_Q	2.1177	(Insuf)	.00000	8.0000	6.0000
Servidor_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Servidor_R Busy	.92929	(Insuf)	.00000	1.0000	1.0000

COUNTERS

Identifier	Count	Limit
Contador Salidas	132	Infinite
Contador Llegadas1	91	Infinite
Contador Llegadas2	48	Infinite

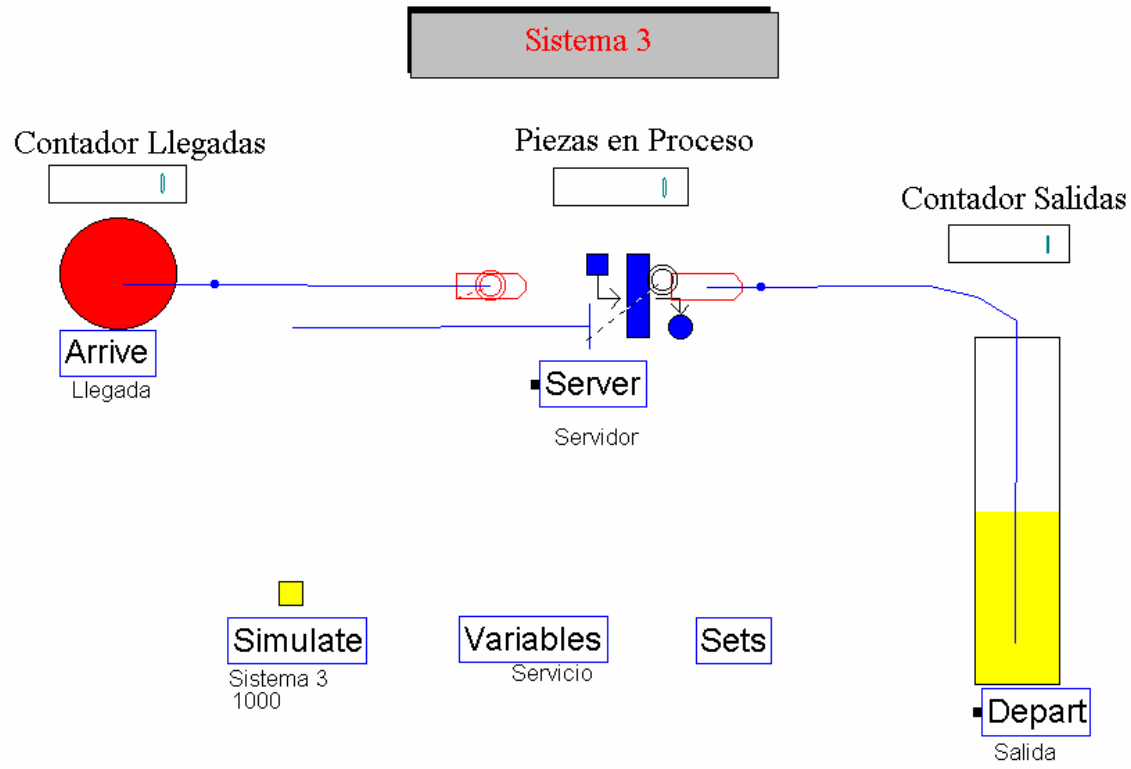
Simulation run time: 0.78 minutes.
Simulation run complete.

Se puede observar que los tiempos medios de permanencia en cola y en el sistema se han reducido significativamente (6.7505 y 12.275 minutos respectivamente) mientras que la carga del servidor sigue siendo la misma, ya que el número de entidades que llegan es el mismo en igual tiempo de simulación. Se comprueba entonces que esta disciplina de cola mejora el comportamiento del sistema de forma considerable.

Caso 3: Población con usuarios distintos.

El sistema que se va a estudiar a continuación es un sistema de colas al que accede una población con una tasa de entrada en el sistema de 5 minutos entre llegadas. Dicha población se sabe que está constituida por dos tipos de usuarios. El primer tipo, que tiene un porcentaje del 80% del total demanda un tiempo de servicio igual a 4 minutos. El segundo tipo, por su parte, demanda 10 minutos de tiempo de servidor. El tiempo que los usuarios tardan en circular entre estaciones es de 1 minuto (*Routes*).

El modelo que se va a utilizar, a pesar de la presencia de dos tipos de usuarios distintos que demandan tiempos distintos de servicio, es el típico de una población, una cola y un servidor.



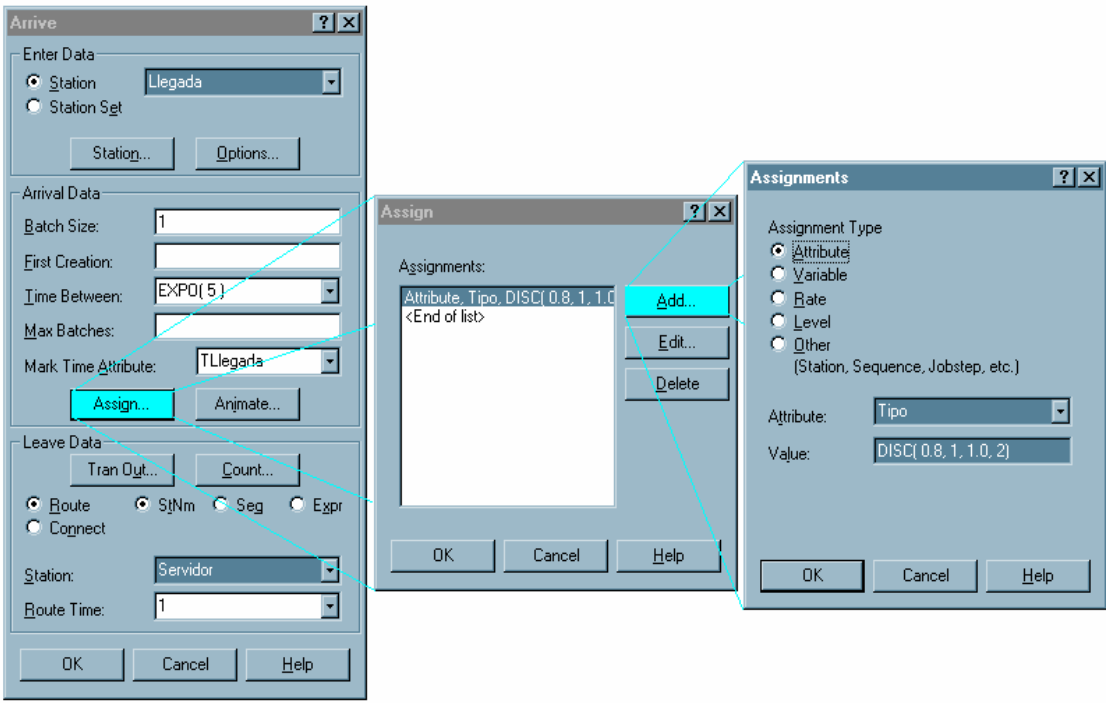
Se puede observar que en este modelo aparecen dos elementos que hacen que difiera del sistema típico de una población, cola y servidor. Estos elementos, los módulos *Variables* y *Sets*, nos permiten realizar la diferenciación entre los dos tipos de usuarios aunque pertenezcan a la misma población.

A continuación se procede al estudio pormenorizado de todos los módulos del sistema.

Módulo Arrive.

Este es el módulo de llegadas. Define cómo van a ser las entidades que llegarán al sistema. Sus tasas, atributos y hacia qué servidor se van a dirigir y cuánto van a tardar.

Para indicar esto se completa el siguiente menú:



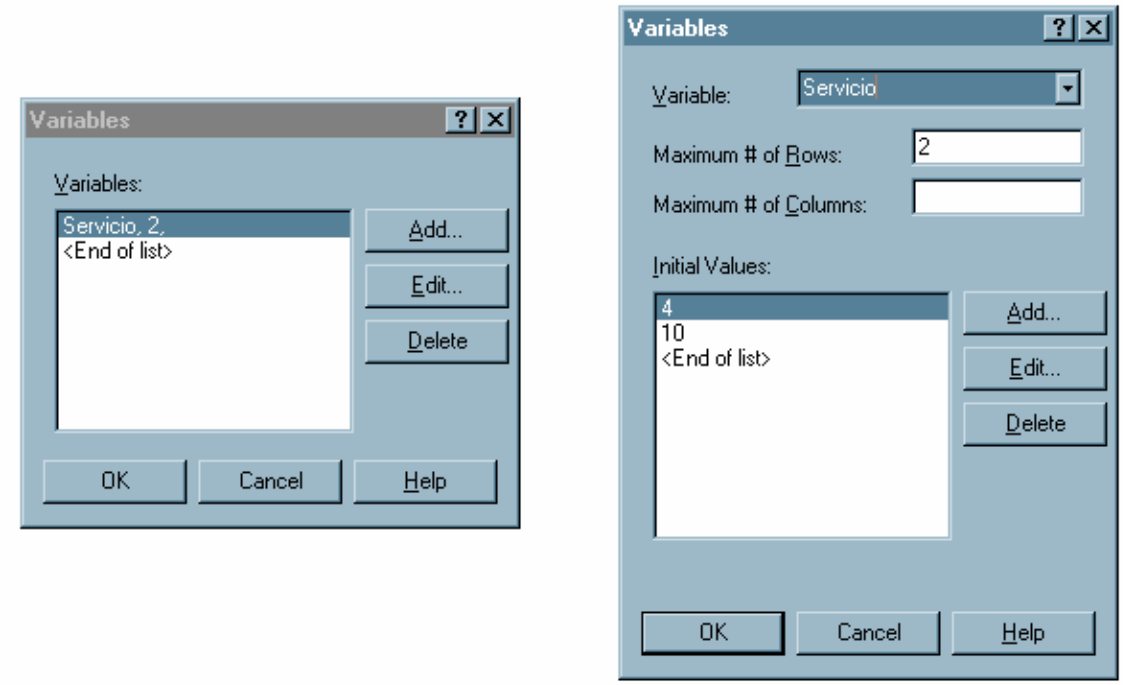
Para distinguir los dos tipos de usuarios se ha asignado a las entidades de la población un atributo que servirá de índice para hacer referencia a cada uno. Este índice, que se ha nombrado Tipo, es una variable DISC que asigna las siguientes probabilidades:

- Tipo 1 → 80%.
- Tipo 2 → 20%.

Hay que tener en cuenta que la variable DISC(a, 1, b, 2, 1, 3) indica que las entidades serán de tipo 1 con probabilidad *a*, de tipo 2 con probabilidad *b-a* y de tipo 3 con probabilidad *1-b-a*. Funciona, por tanto, con probabilidades de forma acumulativa.

Módulo Variables.

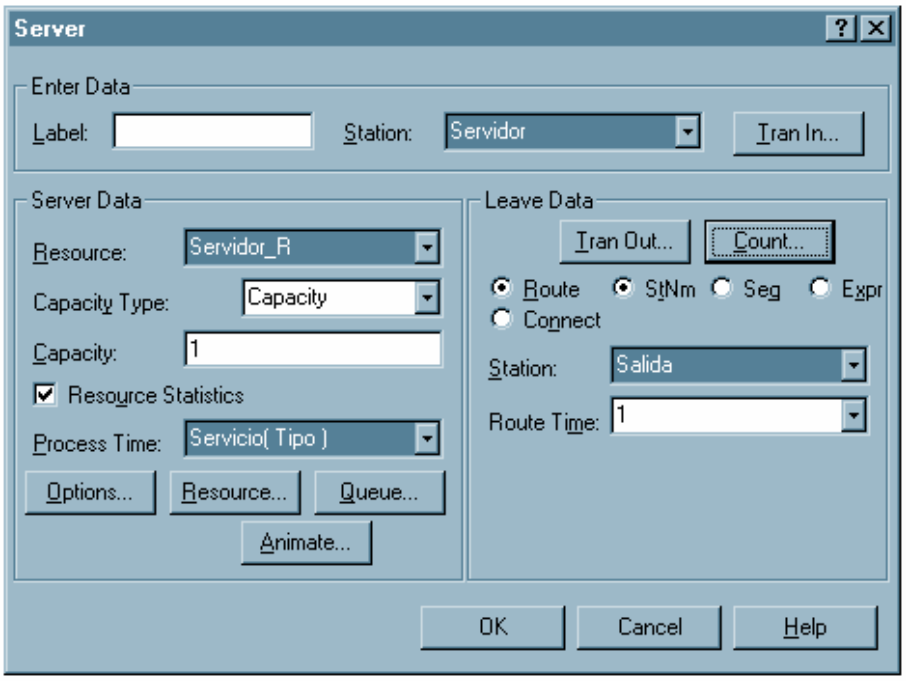
En este módulo se va a definir una variable, Servicio, que representará el tiempo de servicio que solicita cada usuario en el sistema. Como se tienen dos tipos de usuarios la variable será bidimensional y se hará referencia a cada uno de los tiempos de cada entidad mediante el índice Tipo que se definió en el módulo *Arrive*.



Módulo Server.

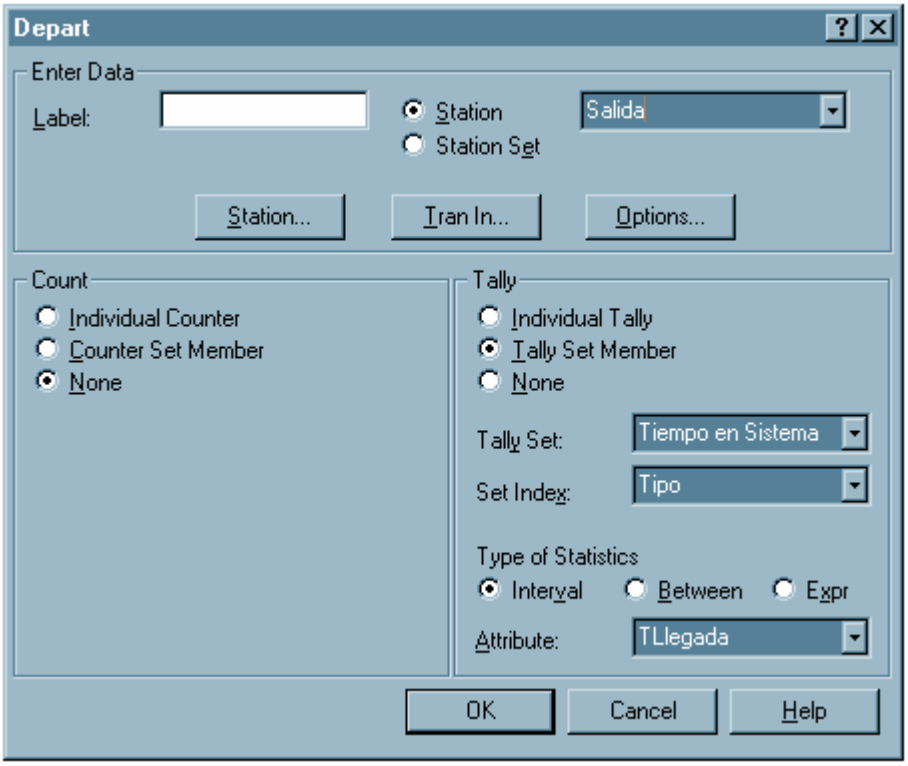
Aquí se define el servidor del sistema. Se va a diseñar como un servidor monoestación, con cola indefinida FIFO.

El tiempo que tarda en servir a las entidades que acceden a él será la *Variable* Servicio indexada por Tipo.



Módulo Depart.

Es el módulo que muestra cómo van saliendo las entidades del sistema.

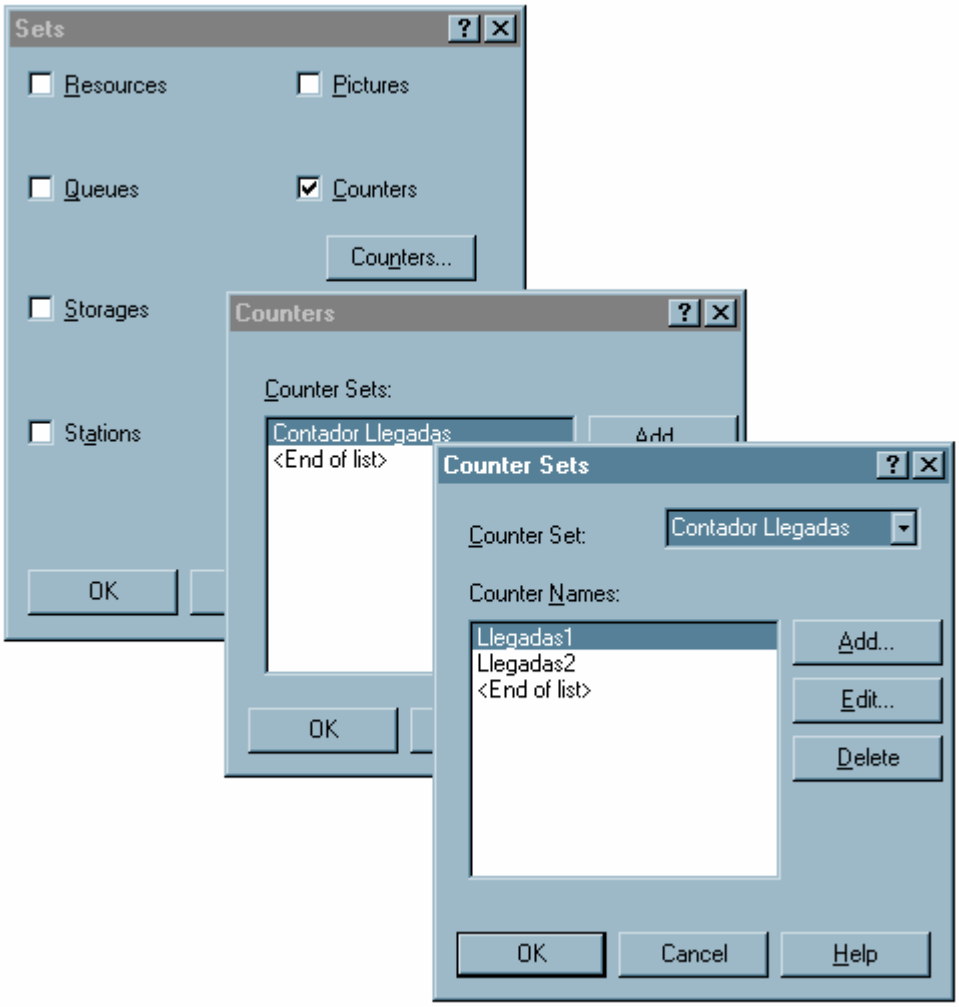


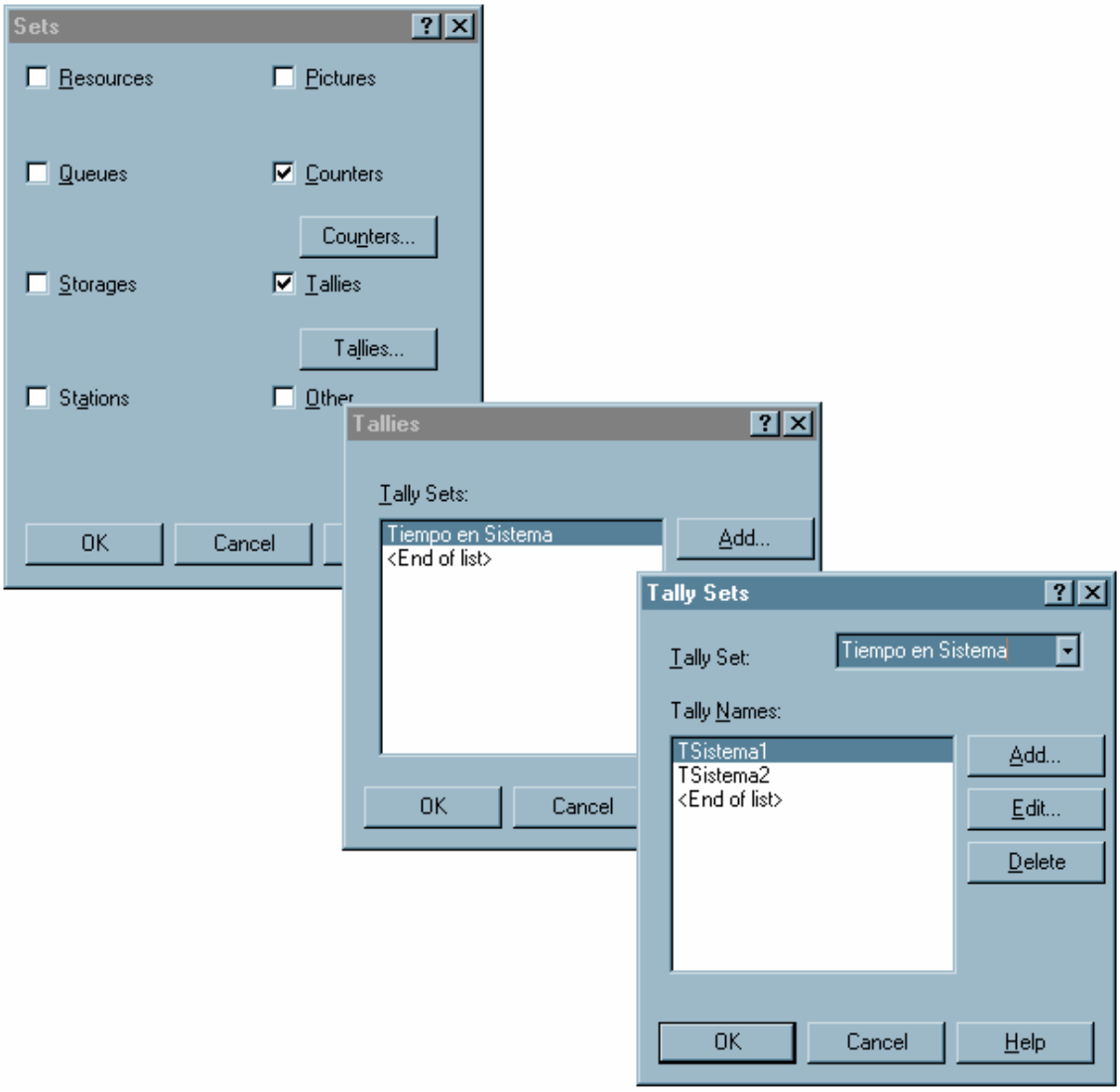
Se puede ver que para obtener el Tiempo en Sistema de las entidades se ha definido un *Tally* que no es simple. Es un *Set* de *Tally*. Esto permite distinguir entre ambos tipos de usuarios. Un *Set* es un grupo de objetos de Arena, en nuestro caso un grupo de *Tallies*. Para hacer referencia al Tiempo en Sistema de cada tipo de entidad se utiliza el índice Tipo. Así se sabrá el tiempo de cada tipo de usuario.

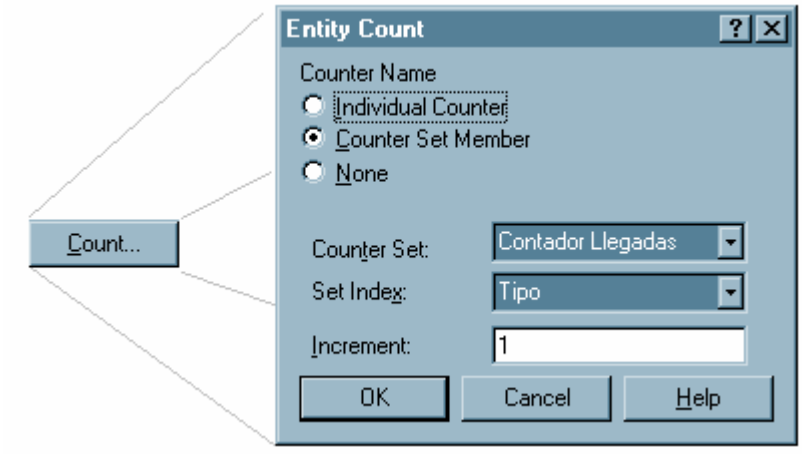
Módulo Sets.

En este módulo se definen los *Sets* que luego se van a utilizar en el resto del sistema.

Aquí se definen los Sets de *Counters* y *Tallies*.

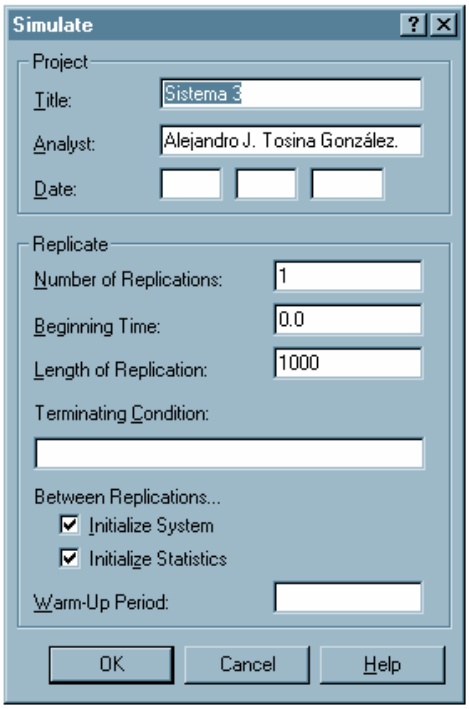



Sets
Resources
Pictures
Queues
Counters
Counters...
Storages
Tallies
Tallies...
Stations
Other
OK
Cancel
Tallies
Tally Sets:
Tiempo en Sistema
<End of list>
Add...
OK
Cancel
Tally Sets
Tally Set:
Tiempo en Sistema
Tally Names:
TSistema1
TSistema2
<End of list>
Add...
Edit...
Delete
OK
Cancel
Help

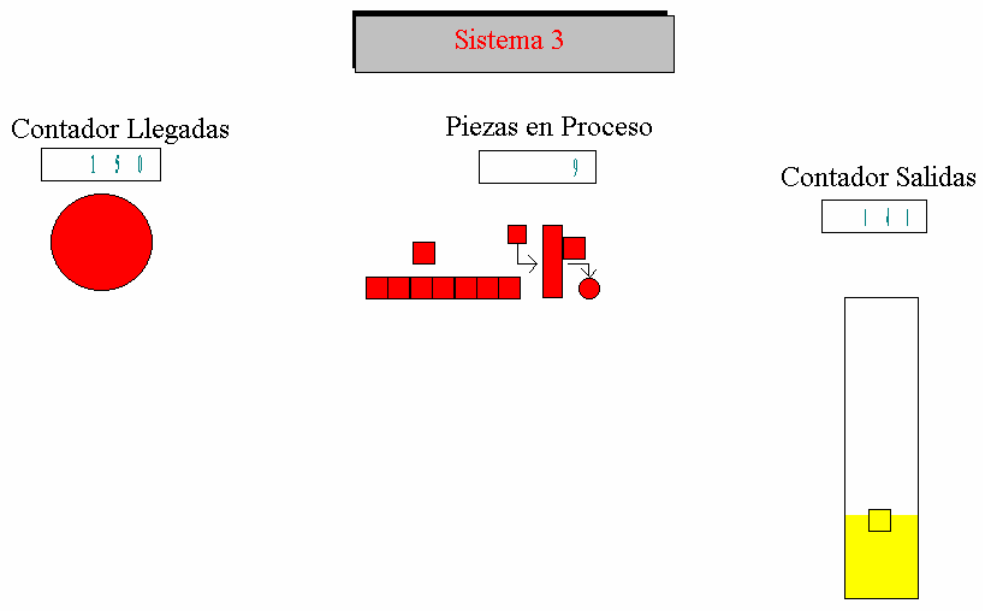

Count...
Entity Count
Counter Name
Individual Counter
Counter Set Member
None
Counter Set:
Contador Llegadas
Set Index:
Tipo
Increment:
1
OK
Cancel
Help

Módulo Simulate.

En este módulo, sólo se va a definir el tiempo de simulación que va a ser utilizado para evaluar este modelo.



Una vez realizado el modelo y verificada su correcta realización se procede a ver los resultados de la simulación.



Los resultados obtenidos son los siguientes:

```
                         ARENA Simulation Results
                  Alejandro Javier Tosina Glez. - License #9400000

                          Summary for Replication 1 of 1

Project:  Sistema 3                            Run execution date :  10/ 1/2000
Analyst:  Alejandro J.Tosi                     Model revision date:  10/ 1/2000

Replication ended at time      : 1000.0

                                  TALLY VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Observations
-------------------------------------------------------------------------------

TSistema1               27.439     (Insuf)    6.0000     71.711        152
TSistema2               28.618     (Insuf)    12.000     75.094         32
Servidor_R_Q Queue Tim  21.025     (Insuf)    .00000     65.711        186

                             DISCRETE-CHANGE VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Final Value
-------------------------------------------------------------------------------

# in Servidor_R_Q       4.1327     (Corr)     .00000     17.000     10.000
Servidor_R Available    1.0000     (Insuf)    1.0000     1.0000     1.0000
Servidor_R Busy         .93239     (Insuf)    .00000     1.0000     1.0000

                                   COUNTERS

                    Identifier                Count   Limit
                    ------------------------------------------

                    Llegadas1                   160   Infinite
                    Llegadas2                    37   Infinite
                    Contador Salidas            185   Infinite

Simulation run time: 0.05 minutes.
Simulation run complete.
```

Se puede apreciar con claridad como han llegado muchas más entidades del Tipo 1 que del Tipo 2. Esto es consecuencia de la definición de probabilidades que se hizo en el módulo *Arrive*. El tiempo de permanencia en el sistema es parecido en ambos usuarios, aunque los de tipo 2 piden más servicio que los de tipo 1. Esto es debido a que la mayor parte del tiempo están en cola, así que la diferencia en tiempos de servicio no resulta significativa en la cifra total.

Caso 4: Comparación de un Sistema de Cola Centralizada con un Sistema Distribuido.

El objetivo de la batería de simulaciones que se va a desarrollar a continuación consiste en establecer una comparativa entre una serie de sistemas que podrían parecer equivalentes para ver cual de ellos será mejor, así como deducir las causas de las diferencias que se verán entre ellos.

Los sistemas que se van a ver son los siguientes:

- 2 M/M/1. Cola FIFO.
- M/M/1 con servidor de doble capacidad que los anteriores.
- M/M/2 con servidores iguales a los del primer caso.

Será interesante comprobar que sistemas de cola centralizada son más eficientes que los de cola distribuida en cuanto a tiempos de permanencia en el sistema puesto que distribuye mejor a los usuarios que llegan al sistema.

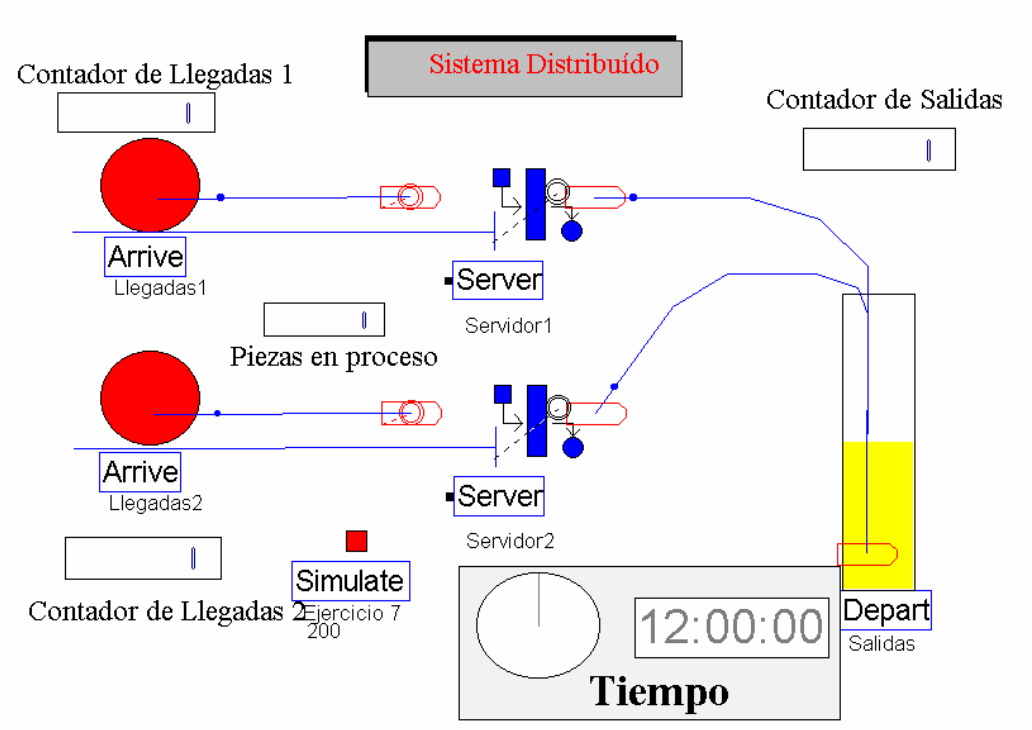
Sistema 2 M/M/1.

El primer sistema que se estudiará consta en realidad de dos subsistemas idénticos M/M/1. Los usuarios llegan a dos colas independientes siguiendo una distribución exponencial con tiempo medio entre llegadas igual a 4 minutos y solicitan al servidor correspondiente un tiempo medio de proceso de media 2 minutos.

La cola es FIFO, por lo que los usuarios de las poblaciones se atenderán por orden de llegada.

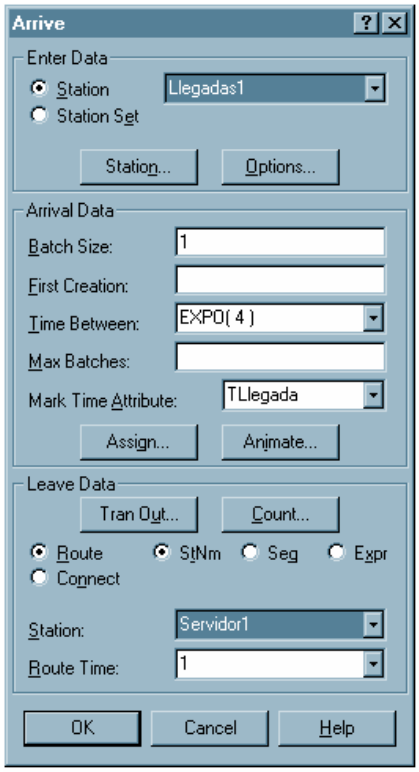
Es necesario resaltar que ambos sistemas de colas son independientes por lo que, aunque un servidor esté libre y en otro haya cola, no habrá traspaso de usuarios de un sistema a otro.

El modelo es el siguiente:

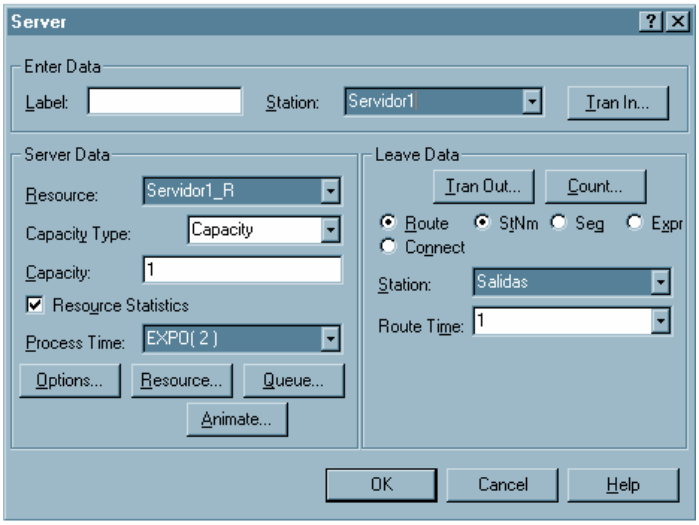


Se ve que se tienen dos fuentes idénticas que piden el mismo servicio a dos servidores idénticos.

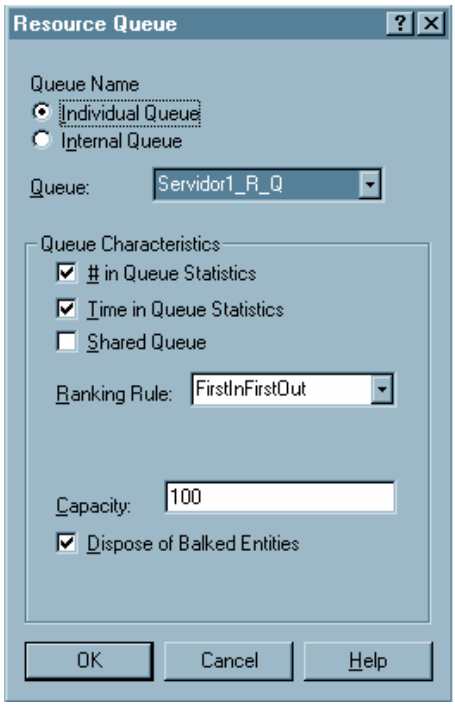
Los parámetros de las fuentes son los siguientes:



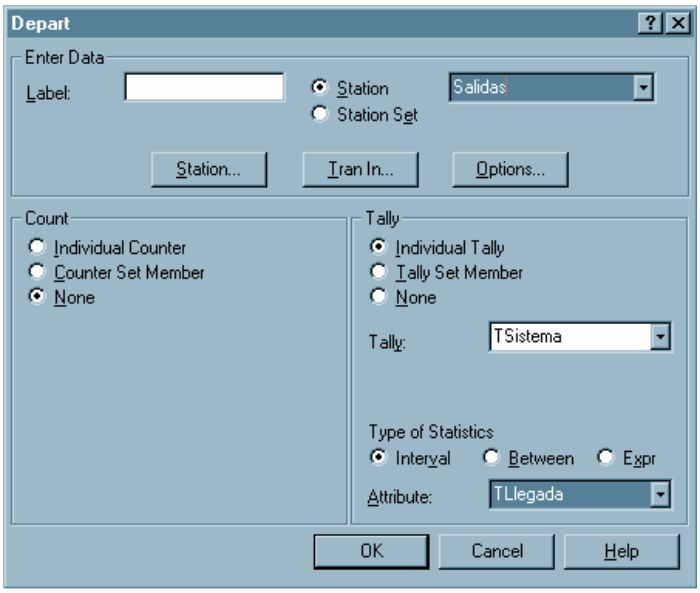
Los parámetros de los servidores, que son monoestación, se describen a continuación:



La cola de este sistema servidor será FIFO y con una capacidad de 100 individuos, suficiente para la simulación.



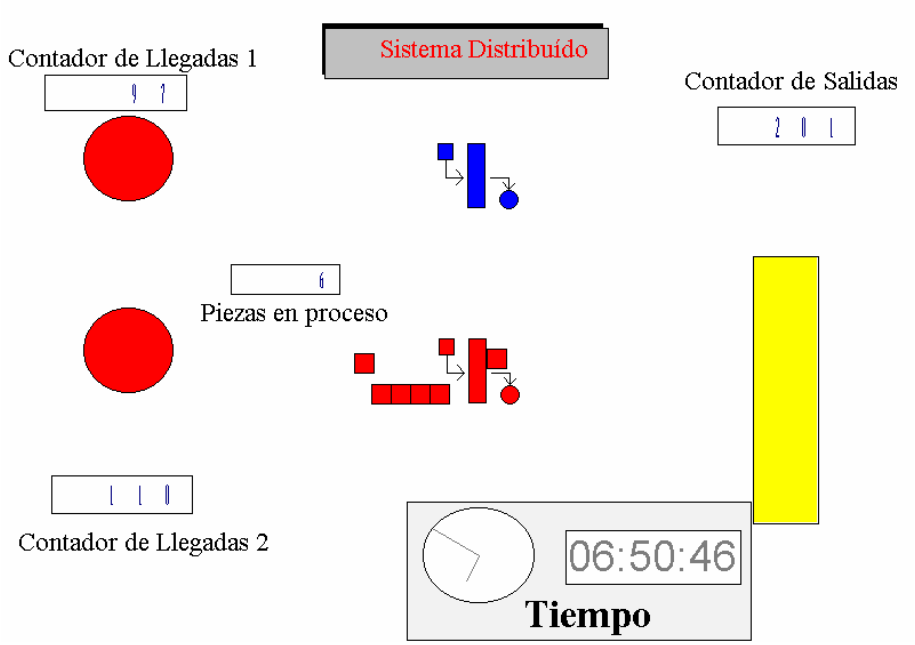
El bloque de salida estará definido de la siguiente forma:



No se define ningún contador asociado al bloque de salida porque será creado aparte.

El *Tally* que se ha establecido permitirá medir el intervalo entre el instante en el que los usuarios llegan al sistema (TLlegada) y el momento en el que lo abandonan.

Una vez establecido el modelo se procede a realizar la simulación durante un intervalo de 500 minutos para ver sus prestaciones.



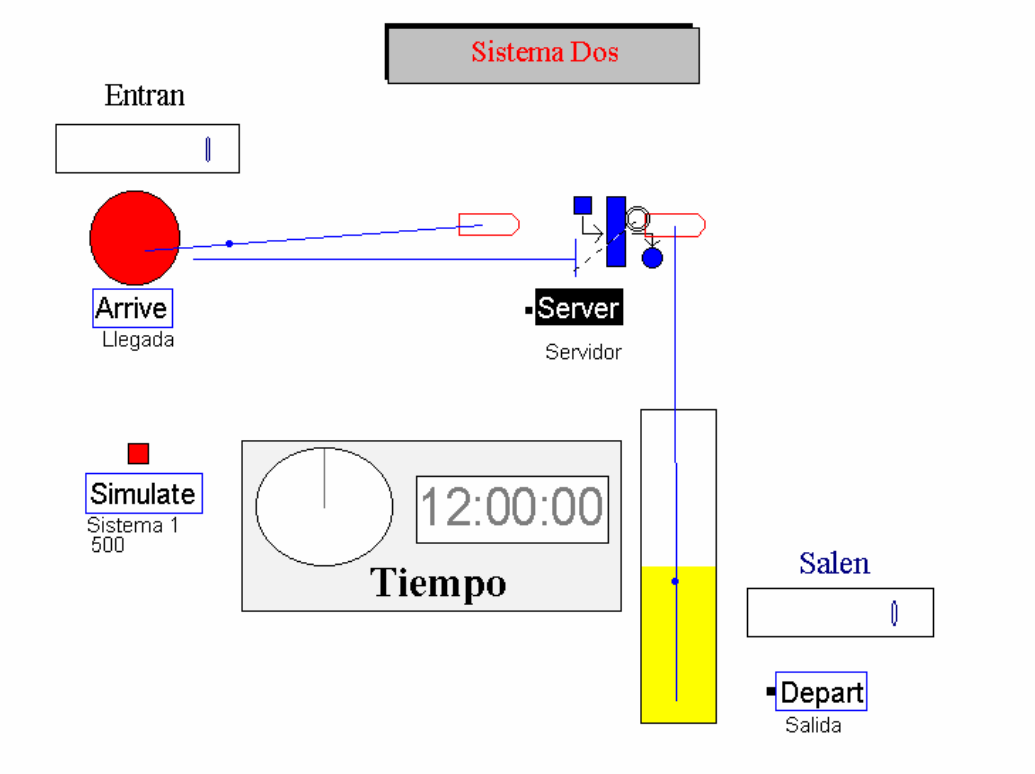
Los resultados de la simulación son los siguientes:

```
                    ARENA Simulation Results
            Alejandro Javier Tosina Glez. - License #9400000

                    Summary for Replication 1 of 1

Project:  Sistema  Distrib                    Run execution date :   9/20/2000
Analyst:  Alejandro Javier                    Model revision date:   8/ 9/2000

Replication ended at time      : 500.0

                                TALLY VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Observations
-------------------------------------------------------------------------------

TSistema                5.7688     (Insuf)    2.0408     19.327        244
Servidor1_R_Q Queue Ti  .94876     (Insuf)    .00000     10.291        117
Servidor2_R_Q Queue Ti  2.5110     (Insuf)    .00000     17.057        129

                           DISCRETE-CHANGE VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Final Value
-------------------------------------------------------------------------------

# in Servidor2_R_Q      .64785     (Insuf)    .00000     6.0000     .00000
# in Servidor1_R_Q      .22201     (Insuf)    .00000     4.0000     .00000
Servidor2_R Busy        .52126     (Insuf)    .00000     1.0000     1.0000
Servidor1_R Busy        .45996     (Insuf)    .00000     1.0000     .00000
Servidor2_R Available   1.0000     (Insuf)    1.0000     1.0000     1.0000
Servidor1_R Available   1.0000     (Insuf)    1.0000     1.0000     1.0000

                                COUNTERS

                    Identifier                Count   Limit
                    ------------------------------------------

                    Contador Salidas1           117   Infinite
                    Contador Llegadas1          117   Infinite
                    Contador Salidas2           128   Infinite
                    Contador Llegadas2          129   Infinite

Simulation run time: 1.22 minutes.
Simulation run complete.
```

Se puede apreciar que, aunque el tiempo de permanencia en el sistema ha sido muy pequeño, 5.7688 minutos, el tiempo de permanencia en cola muestra unos resultados muy dispares pues en la primera cola es de 0.94876 minutos mientras que en la segunda es de 2.5110 minutos. Esto demuestra lo que se comentó anteriormente. Este sistema es poco eficiente porque puede haber usuarios esperando en una cola mientras que el otro servidor está desocupado.

Sistema M/M/1 de doble capacidad.

Este sistema constará de una población que llegará con tasa doble de la anterior a un servidor con cola FIFO pero que tardará la mitad de tiempo en dar el servicio requerido.



El modelo es sencillo, de una cola y un servidor. Los resultados que obtenemos son los siguientes:

```
                    ARENA Simulation Results
          Alejandro Javier Tosina Glez. - License #9400000

                  Summary for Replication 1 of 1

Project:  Sistema Dos                      Run execution date :   9/20/2000
Analyst:  Alejandro Javier                 Model revision date:   9/ 8/2000

Replication ended at time      : 500.0
```

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
Servidor_R_Q Queue Tim	.92452	(Insuf)	.00000	7.5932	259
Tsistema	3.9661	(Insuf)	2.0038	9.7009	257

DISCRETE-CHANGE VARIABLES

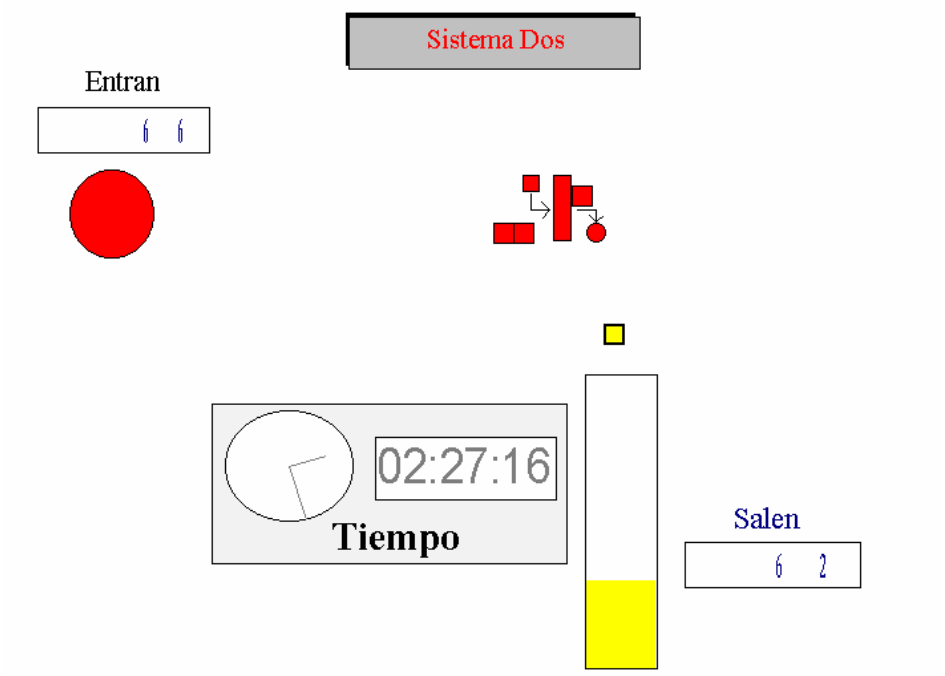
Identifier	Average	Half Width	Minimum	Maximum	Final Value
# in Servidor_R_Q	.48200	(Insuf)	.00000	6.0000	1.0000
Servidor_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Servidor_R Busy	.54032	.08623	.00000	1.0000	1.0000

COUNTERS

Identifier	Count	Limit
Salen	257	Infinite
Entran	260	Infinite

Simulation run time: 1.75 minutes.
Simulation run complete.

Este sistema es más eficiente que el anterior puesto que los tiempos de permanencia en cola y en el sistema (0.92452 y 3.9661 minutos respectivamente) son menores que antes, así como la carga del servidor es algo mayor, con lo cual se le saca más provecho que antes.

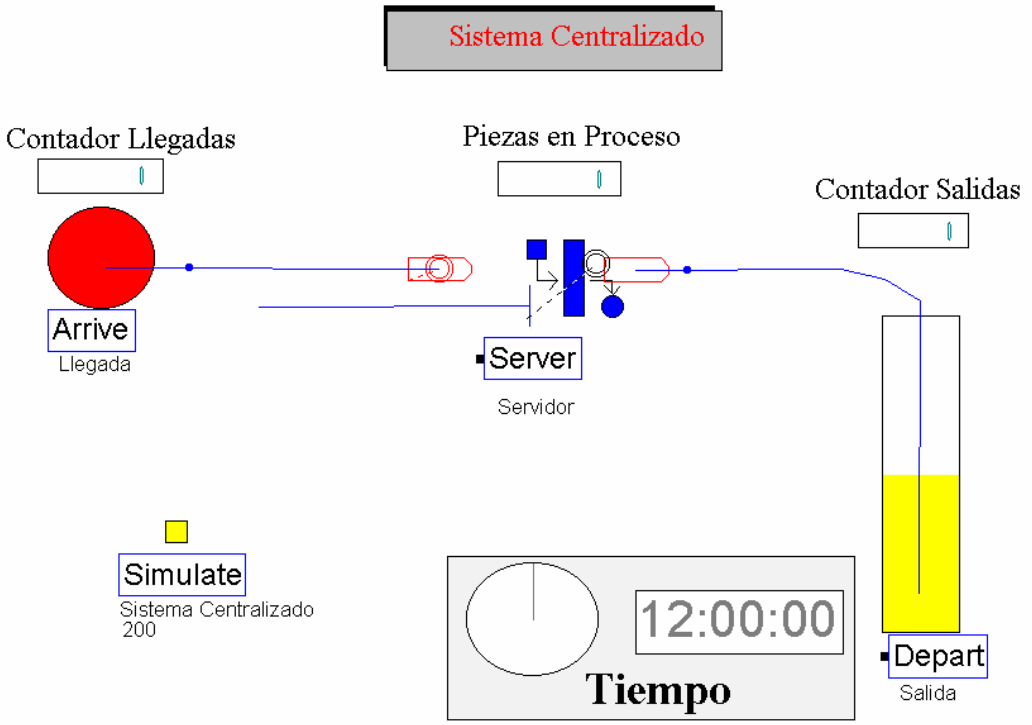


Sistema M/M/2.

Este sistema tiene dos servidores idénticos a los que se accede por una única cola con disciplina FIFO.

Los usuarios de la población llegan con una tasa exponencial de media 2 minutos y solicitan un servicio de duración media de 2 minutos también. Así se ve que todos los sistemas comentados son equivalentes.

El modelo desarrollado es el siguiente:



Se muestran a continuación los resultados de la simulación:

```
                    ARENA Simulation Results
            Alejandro Javier Tosina Glez. - License #9400000

                    Summary for Replication 1 of 1

Project:  Sistema Centrali                    Run execution date :   9/20/2000
Analyst:  Alejandro J.Tosi                    Model revision date:   9/20/2000

Replication ended at time      : 200.0

                                TALLY VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Observations
---------------------------------------------------------------------------

Servidor_R_Q Queue Tim  .54062     (Insuf)    .00000     6.2660         86
TSistema                4.3860     (Insuf)    2.0176     10.995         85

                           DISCRETE-CHANGE VARIABLES

Identifier              Average   Half Width  Minimum    Maximum   Final Value
---------------------------------------------------------------------------

# in Servidor_R_Q       .23247     (Insuf)    .00000     4.0000     .00000
Servidor_R Available    2.0000     (Insuf)    2.0000     2.0000     2.0000
Servidor_R Busy         .78723     (Insuf)    .00000     2.0000     .00000

                                COUNTERS

                    Identifier                Count   Limit
                    ---------------------------------------

                    Contador Llegadas            86  Infinite
                    Contador Salidas             86  Infinite

Simulation run time: 0.32 minutes.
Simulation run complete.
```

En esta simulación se puede comprobar que el último sistema es el más eficiente. El tiempo medio de permanencia en cola es el menor de los tres sistemas estudiados, 0.54062 minutos.

Este sistema de cola centralizada es mejor que el primer sistema de cola distribuida porque no desperdicia capacidad. Ningún servidor está desocupado mientras haya usuarios esperando ser atendidos en la cola.

También es mejor que el segundo sistema que se estudió, aunque tengan la misma capacidad nominal, porque ahora los procesos largos no monopolizan el servidor ya que se tienen dos trabajando en paralelo.

Caso 5: Estudio de un sistema de fabricación utilizando técnicas de Secuenciación.

El objetivo de este caso es analizar un sistema sencillo de fabricación, haciendo uso de los instrumentos de secuenciación de tareas que nos proporciona Arena 3.0.

El sistema que se va a modelar consta de una estación de entrada, a partir del cual las entidades iniciarán su andadura por el sistema, cuatro estaciones de procesamiento y una estación de salida.

Las estaciones de procesamiento 1,2 y 4 tienen una sola máquina mientras que la estación 3 dispone de dos máquinas distintas. Una de esas máquinas de la estación 3 es más moderna que la otra y puede procesar las entidades que llegan (que llamaremos partes) a una velocidad que es un 80% superior a la disponible por la máquina más antigua.

Los pasos que siguen las distintas partes y lo que tardan en cada estación se definen en la siguiente tabla (los tiempos son distribuciones triangulares).

Tipo de Parte	Estación Tiempo	Estación Tiempo	Estación Tiempo	Estación Tiempo	Estación Tiempo
1	1 TRIA(6,8,10)	2 TRIA(5,8,10)	3 TRIA(15,20,25)	4 TRIA(8,12,16)	
2	1 TRIA(11,13,15)	2 TRIA(4,6,8)	4 TRIA(15,18,21)	2 TRIA(6,9,12)	3 TRIA(27,33,39)
3	2 TRIA(7,9,11)	1 TRIA(7,10,13)	3 TRIA(18,23,28)		

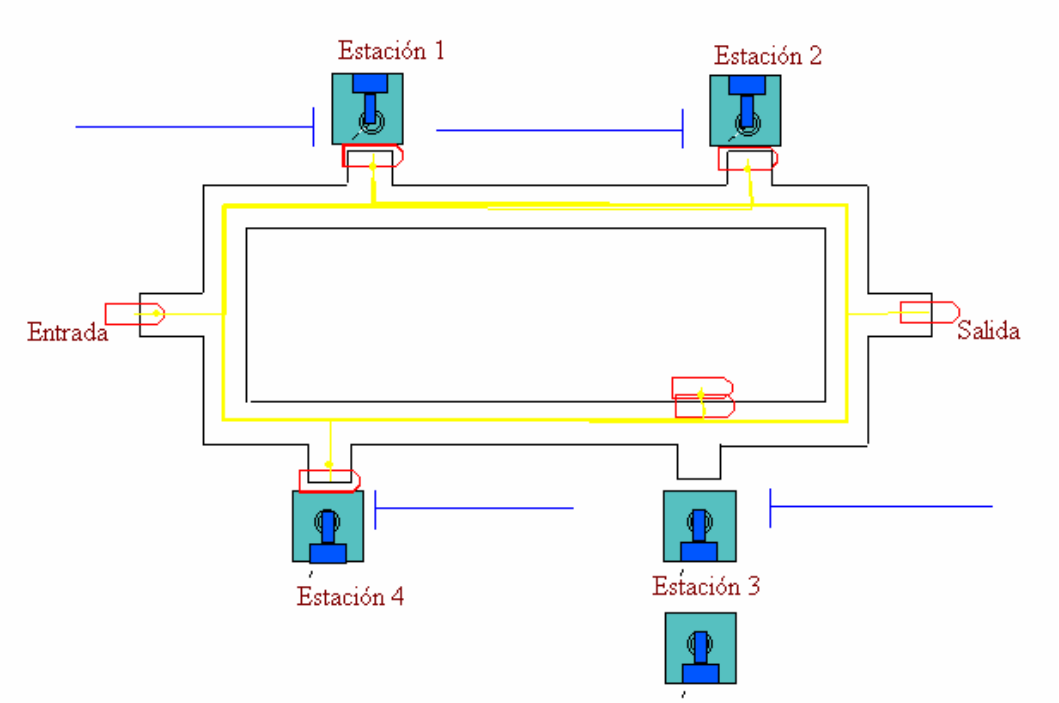
Las partes van llegando al sistema con una distribución exponencial de media 13 minutos entre llegadas (distribución de todas las partes combinadas). Los distintos tipos de partes se distribuyen de la siguiente forma:

- Parte 1 → 26%.
- Parte 2 → 48%.
- Parte 3 → 26%.

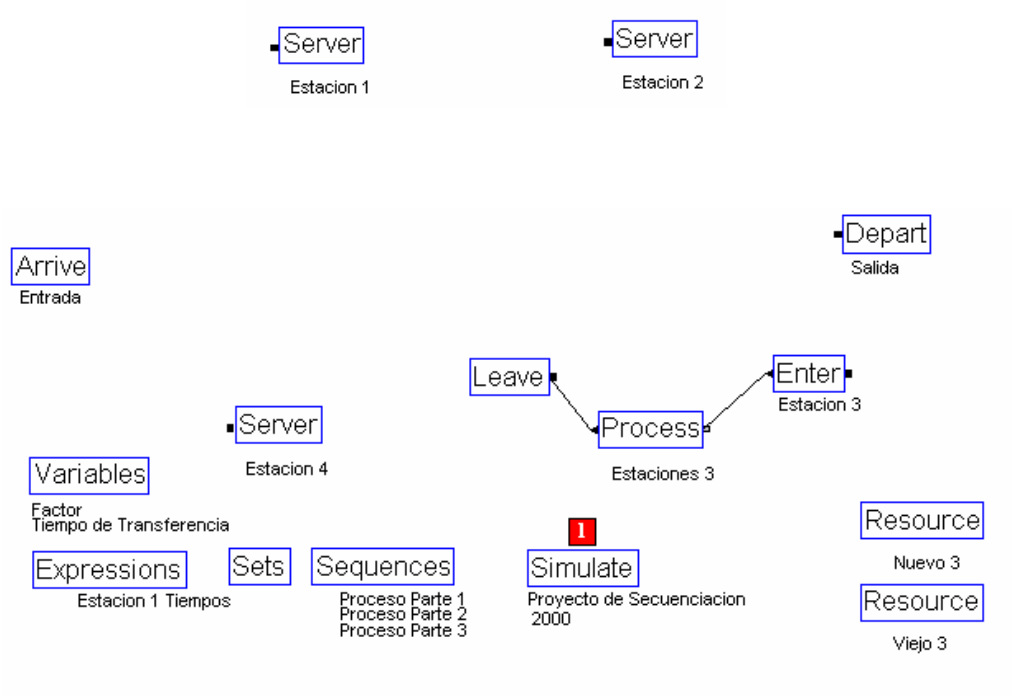
En la figura se muestra el recorrido de las partes por el sistema. Estas partes entran por la izquierda, salen por la derecha y se mueven solamente en la dirección de las agujas del reloj. El tiempo entre estaciones es de 2 minutos.

Se desea obtener estadísticas de utilización de los recursos, información sobre las colas y tiempos de permanencia total en el sistema para cada parte. El tiempo de simulación se establecerá en 2000 minutos.

Con todas estas premisas el sistema en cuestión es el siguiente:



El modelo lógico de este sistema es el siguiente.



Este esquema empleado se verá a continuación de forma pormenorizada para entender cada uno de sus bloques y comprender, de esta forma, cómo ha sido realizado.

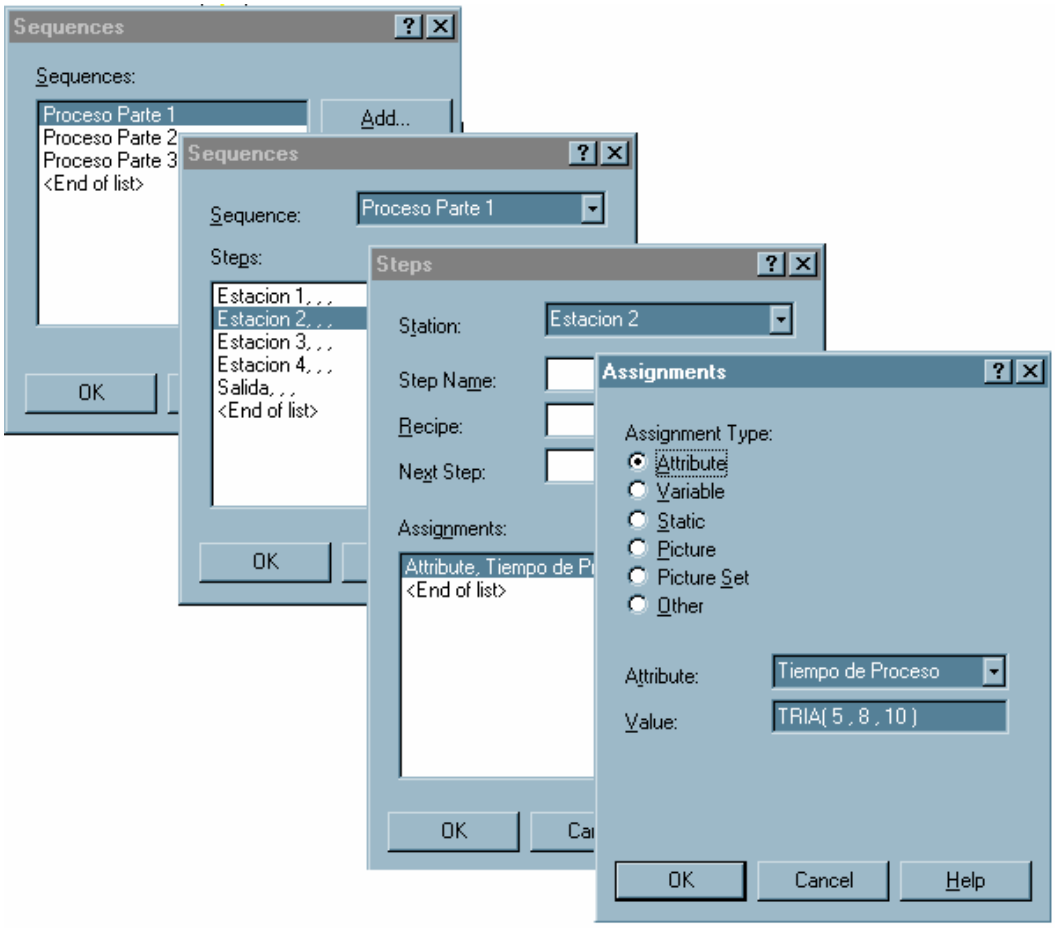
Módulo Sequences.

Las entidades que entran en el sistema seguirán una determinada secuencia de estaciones hasta que sean procesadas completamente. Estas secuencias se definen en el módulo *Sequences*.

Será necesario definir tres secuencias distintas, una para cada tipo de parte, pues cada una seguirá una secuencia de estaciones distinta y con unos tiempos de procesado distintos en cada estación.

Estas secuencias se definen de la siguiente forma:

Sequences	
Sequence	Proceso Parte 1
Steps	
Station	Estacion 2
Assignments	
Assignment Type	
Attribute	select
Attribute	Tiempo de Proceso
Value	TRIA(5,8,10)



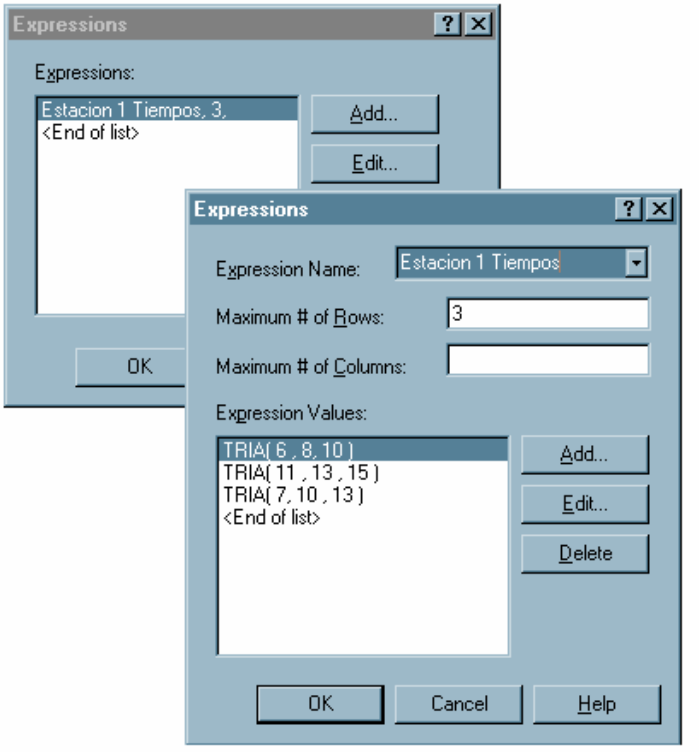
De esta forma se establecen todas las secuencias de estaciones que van a seguir las partes en el sistema.

El tiempo de proceso de las partes en la Estación 1 no se ha especificado en las secuencias. Esto es así porque se hará de otra forma equivalente con el bloque *Expressions*.

Módulo Expressions.

Este módulo se encarga de asociar a un nombre una expresión matemática. Cuando se hace referencia al nombre en cuestión, la expresión asociada es evaluada y se devuelve su valor.

En el presente caso se va a utilizar este módulo para especificar el tiempo que tardarán las partes en ser procesadas en la Estación 1. Será una expresión de dimensión 3 porque hay 3 partes distintas.

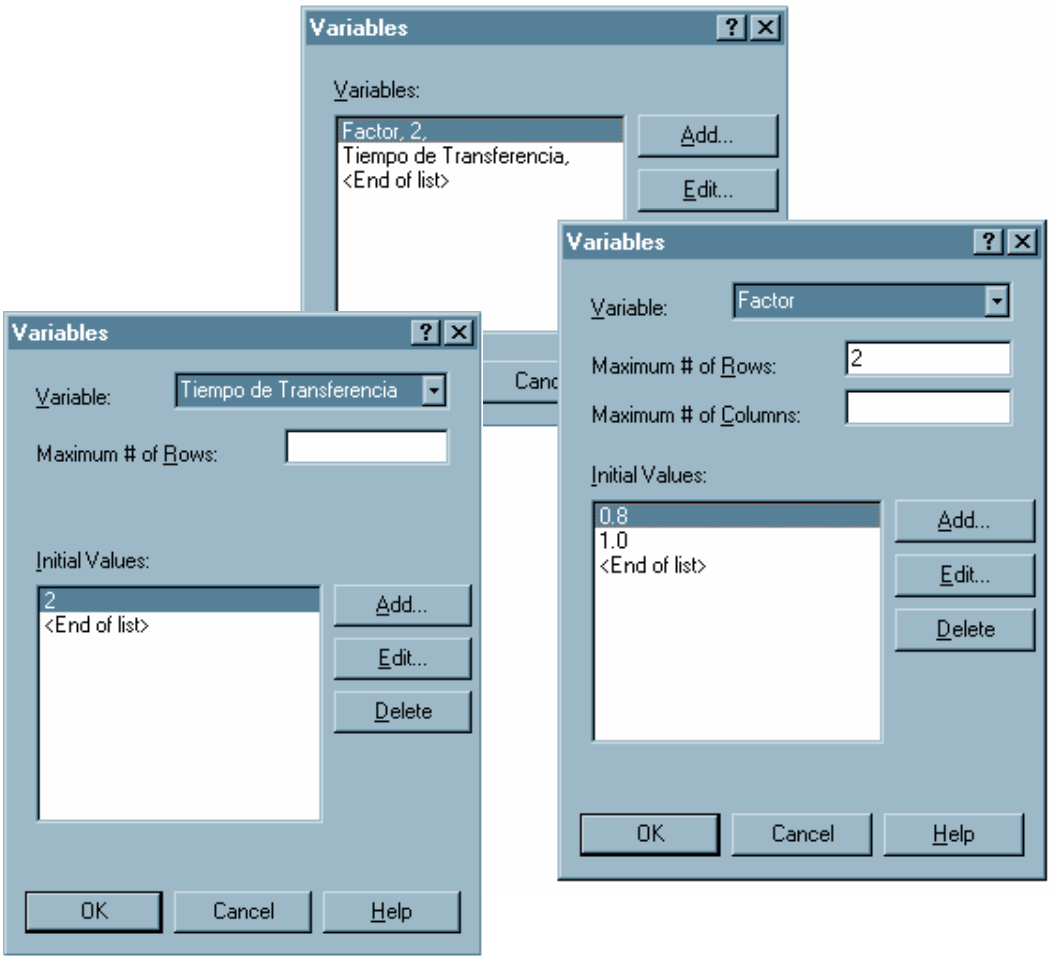


Módulo Variables.

Hay que hacer observar que los tiempos que se han fijado en la Estación 3 de cada secuencia se corresponden al proceso en la máquina antigua. En la nueva es un 20% menor. Esto se modelará con un factor que multiplicará al tiempo de proceso en esta máquina a través de una *Variable* bidimensional, que valdrá 1 para la máquina antigua y 0.8 para la máquina nueva. El valor concreto se obtiene indexando esta variable bidimensional mediante un índice.

Otra variable que se va a definir es el Tiempo de Transferencia, es decir, el tiempo que tardará una parte en ir de una estación a otra a través de un *Route*. Esto se hace así por si se desea considerar este tiempo como parámetro de la simulación.

Estas variables se definen de la siguiente forma:

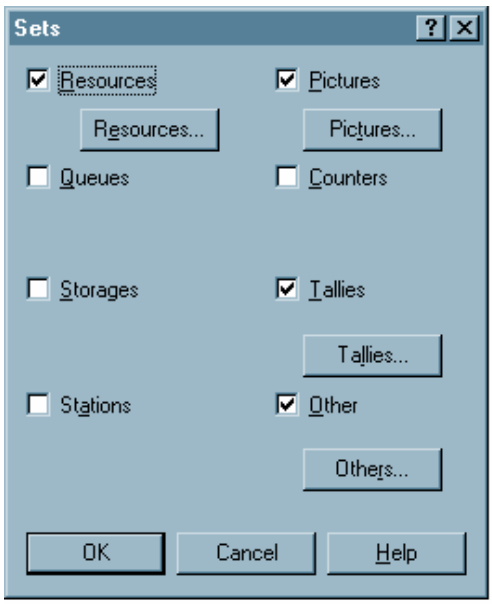


Módulo Sets.

Este módulo permite definir grupos de objetos iguales de Arena tales como *Resources, Storages, Stations*, etc... El acceso a cada elemento del *Set* se hará mediante un índice.

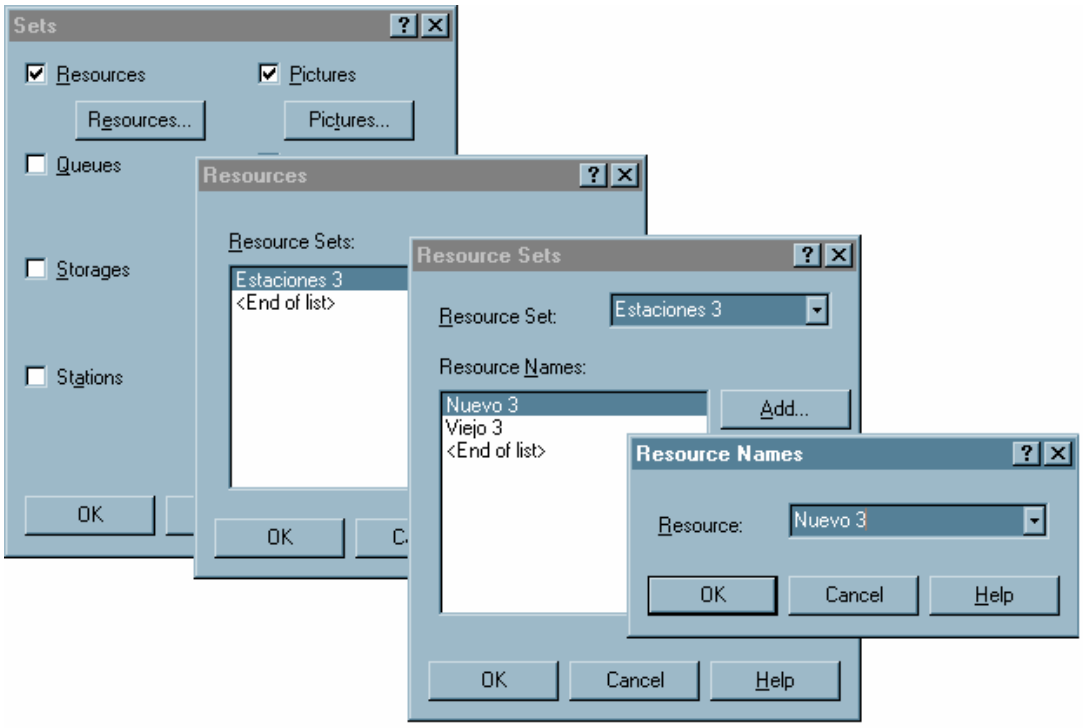
Se va a utilizar este bloque para definir grupos de:

- *Resources*: Serán las máquinas de la Estación 3. Hay que tener en cuenta que se va a establecer que la máquina nueva sea la número 1 por lo que se definirá en primer lugar.
- *Pictures*: Se definen los distintos dibujos que representarán a las partes en la simulación.
- *Tallies*: Miden un tiempo de permanencia en el sistema para cada parte por separado, en vez de medir el intervalo común.
- *Other*: Definen un grupo de *Sequences* para luego asignarle a cada parte su secuencia correspondiente.

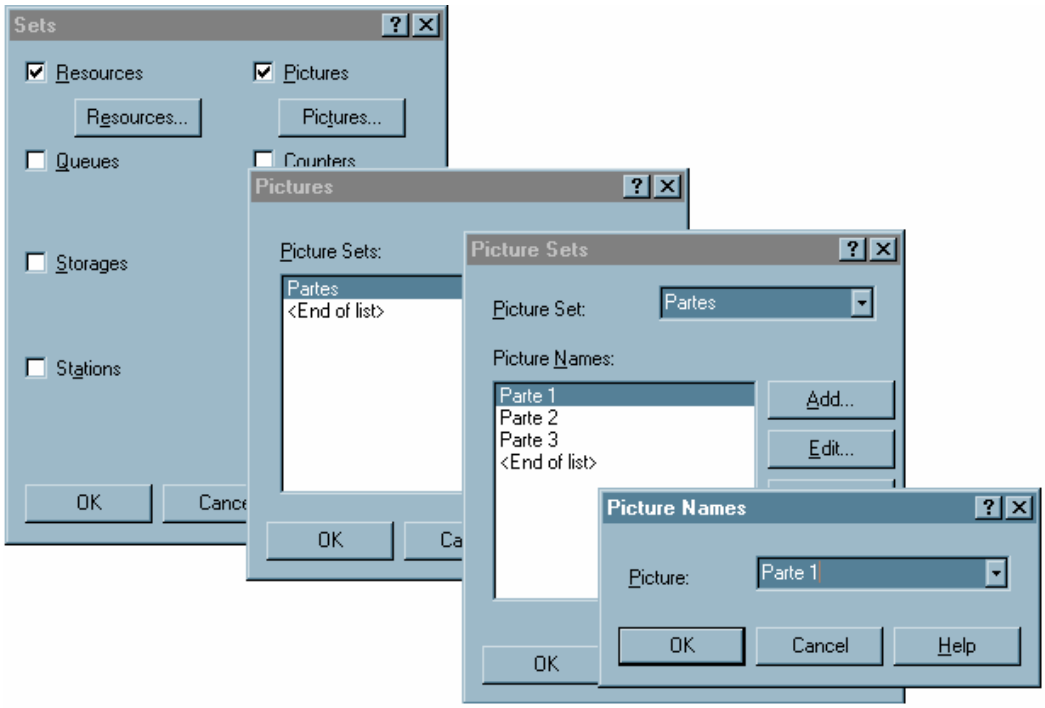


A continuación se muestra de forma más completa cómo se haría con los *Resources*. Con los demás elementos se hará igual. Las dos máquinas se denominarán:

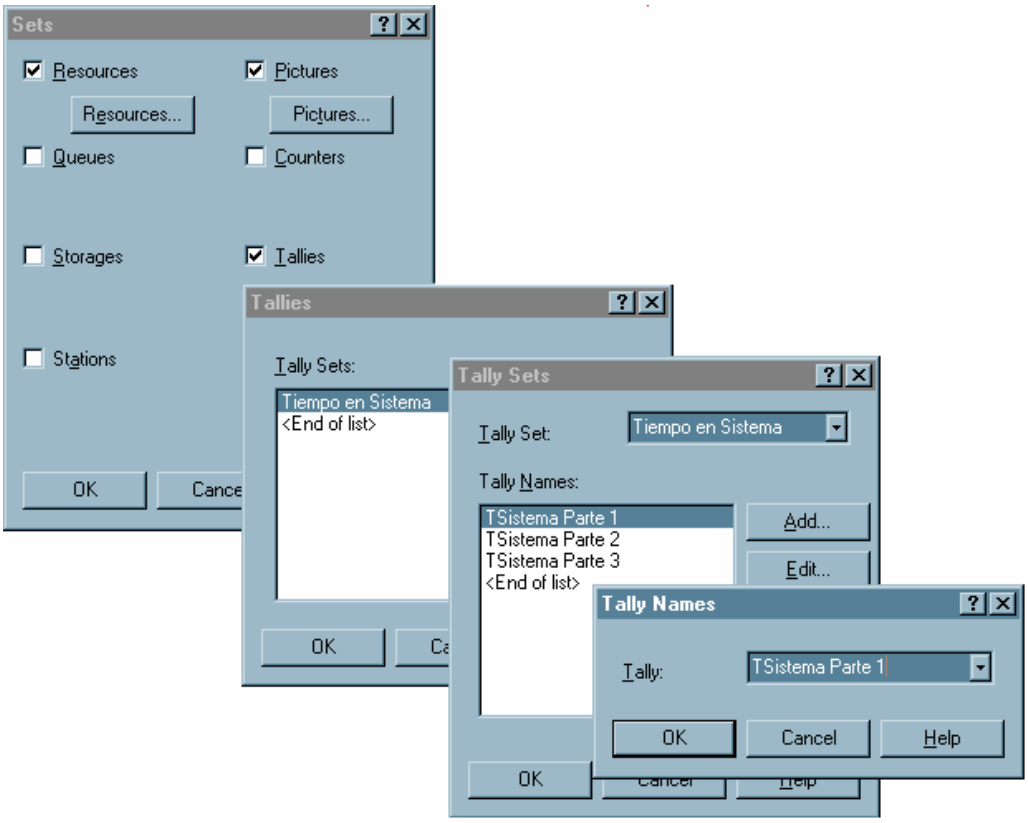
- Nuevo 3.
- Viejo 3.



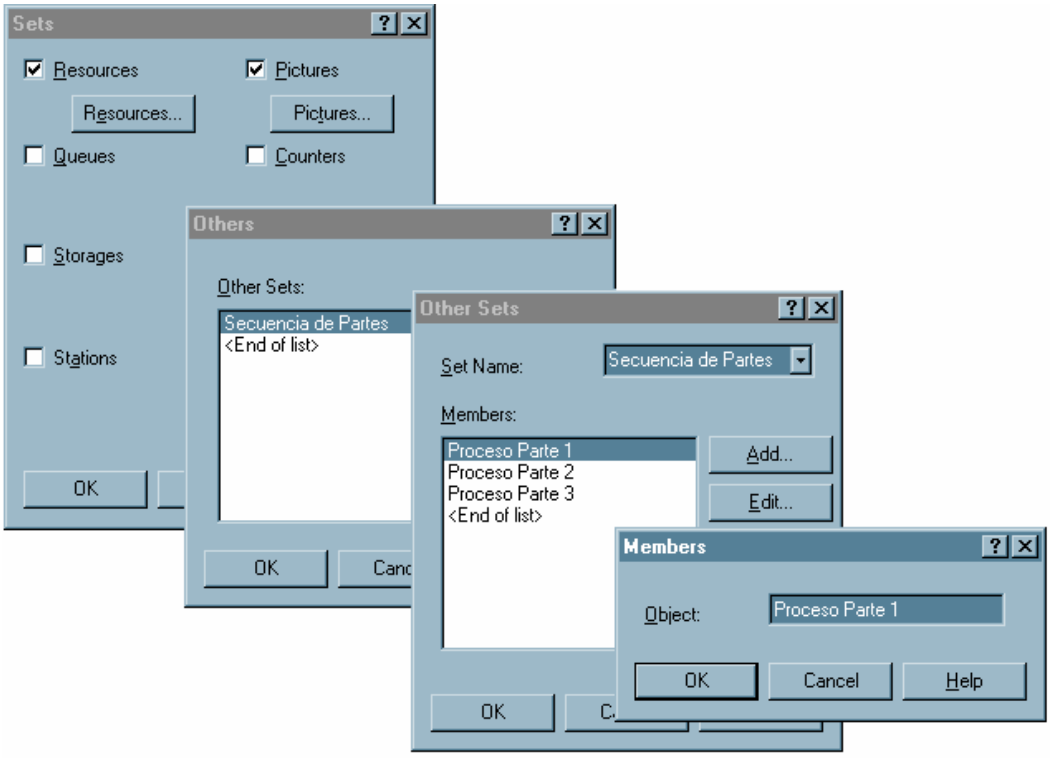
El *Set* de *Pictures* será:



El *Set* de *Tallies* se define de la siguiente forma:

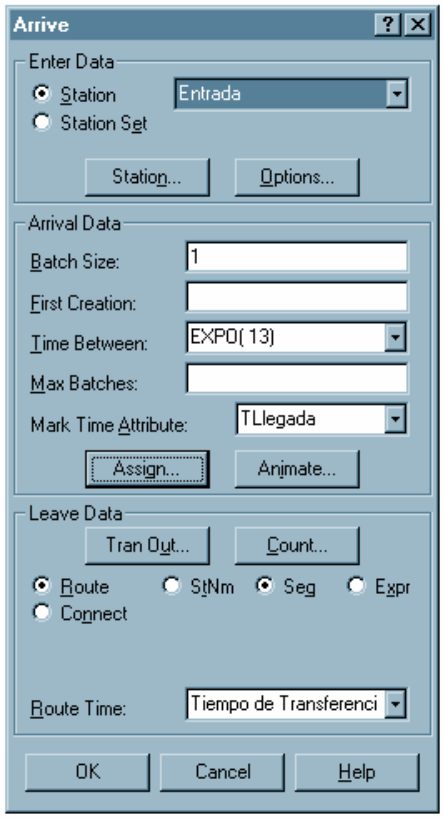


El *Set* de *Sequences* es de la forma:



Módulo Arrive.

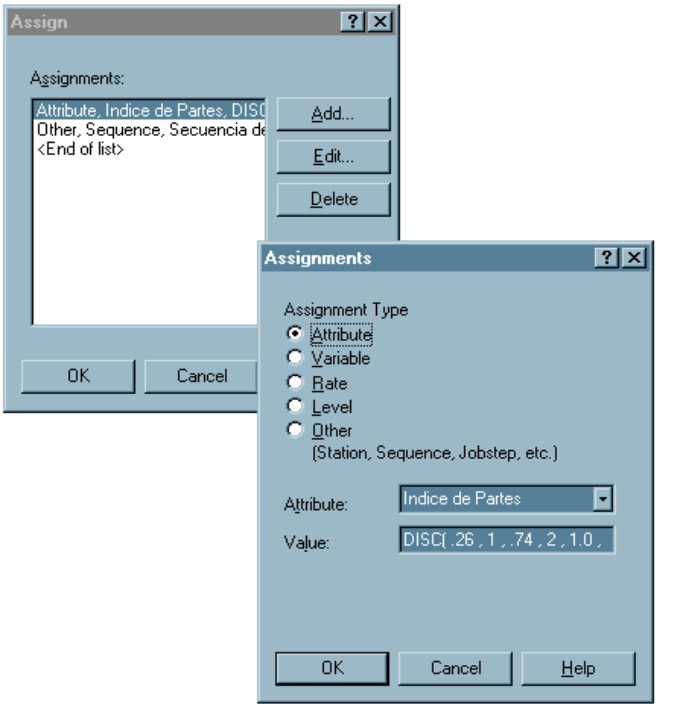
Es el módulo de llegadas al sistema. Define la forma en la que las partes van a entrar en nuestro sistema de producción, su distribución y su tipo.



Como se puede observar, el tiempo entre llegadas es una variable aleatoria exponencial de media 13 minutos. Se marcará un atributo TLlegada para que a la salida se pueda medir el tiempo total de permanencia en el sistema para cada parte.

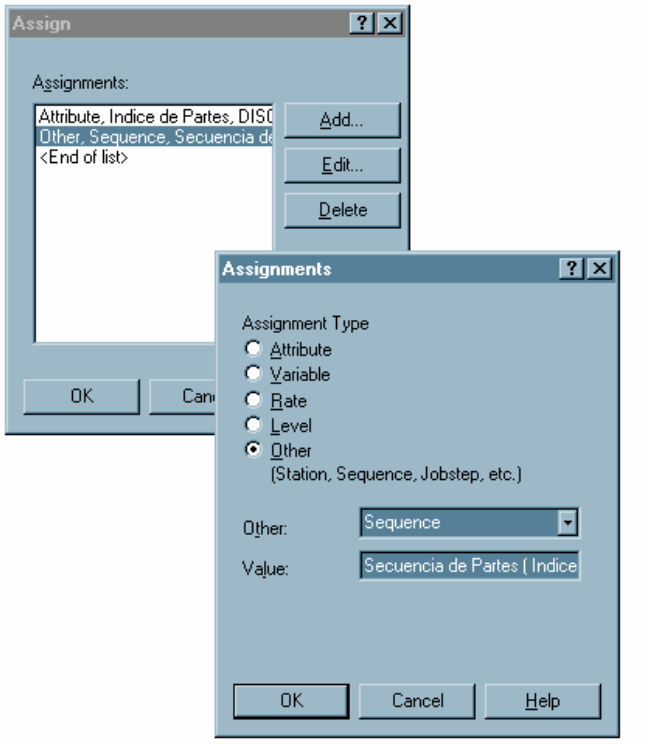
En el submenú *Assign* se van a definir dos atributos que serán necesarios en nuestro modelo:

- Indice de Partes: Marca a cada parte con el tipo al que pertenece. Servirá como índice en las variables y expresiones tridimensionales que se tienen. Es una variable aleatoria DISCRETE que asigna las siguientes probabilidades:
    - Parte 1 → 26%.
    - Parte 2 → 48%.
    - Parte 3 → 26%.

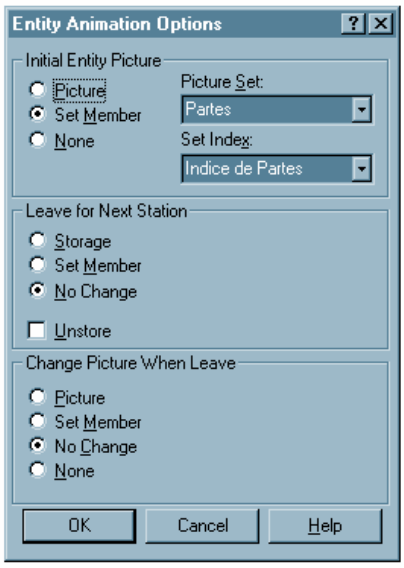


- Secuencia que se le asigna. Cuando una entidad llega, el módulo *Arrive* le asigna a un tipo de parte determinado el Indice de Partes y le indica la secuencia que tiene que seguir. Se le asigna la secuencia, definida en el *Set*, que salga de evaluar la siguiente expresión:

```
Secuencia de Partes( Indice de Partes)
```



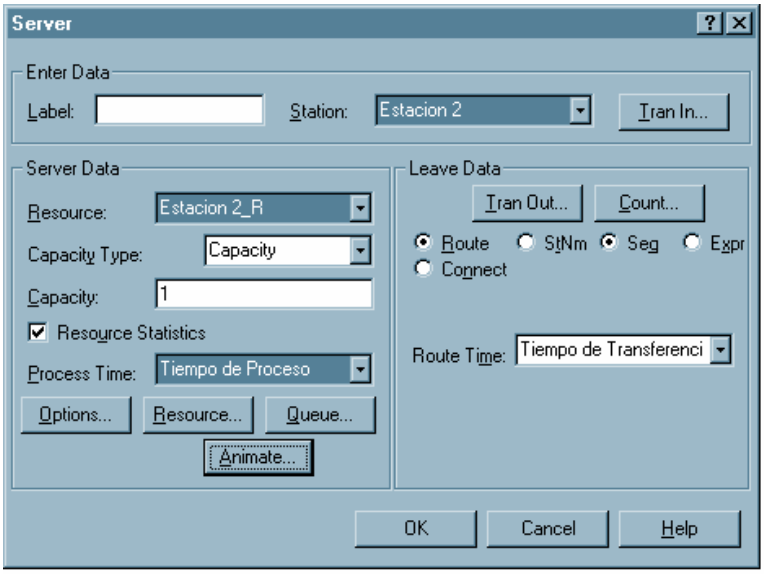
En el apartado *Animate* se define el *Picture* que se asigna a cada parte en cuestión. Para esto también se utiliza el atributo Indice de Parte y el *Set* correspondiente.



De esta forma, el módulo *Arrive* crea las partes, determina su tipo, les asigna la secuencia apropiada y el dibujo correcto y, automáticamente, las encamina hacia la siguiente estación de su secuencia.

Módulos Server.

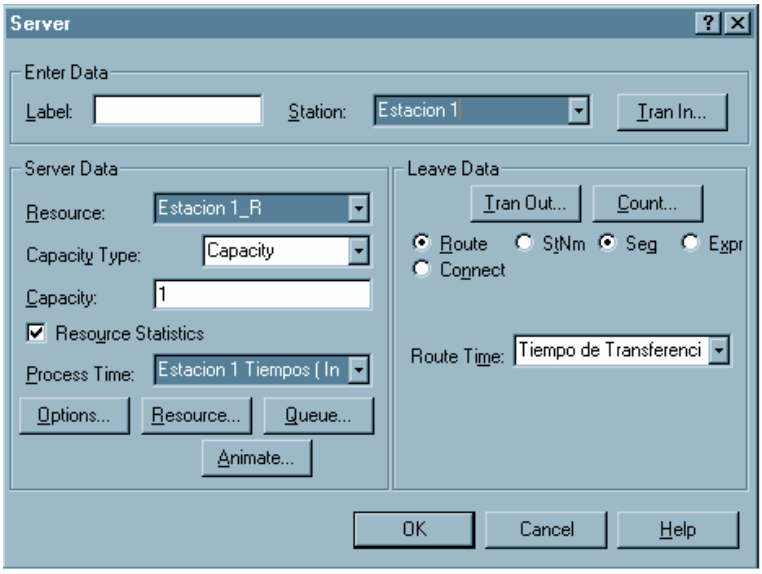
Las estaciones se modelan utilizando este tipo de módulos. Para definir las estaciones 2 y 4 se utiliza el atributo Tiempo de Proceso definido en el módulo *Sequences*. Las entidades salen, después de ser atendidas, a la siguiente estación mediante un *Route* siguiendo la secuencia correspondiente:



Las partes llegan a una cola indefinida de disciplina FIFO y son atendidas con un Tiempo de Proceso que se definió en el módulo *Sequences*.

La Estación 1 cambia su tiempo de proceso. Ahora es una *Expression* indexada por el Indice de Partes.

```
Estacion 1 Tiempos( Indice de Partes)
```

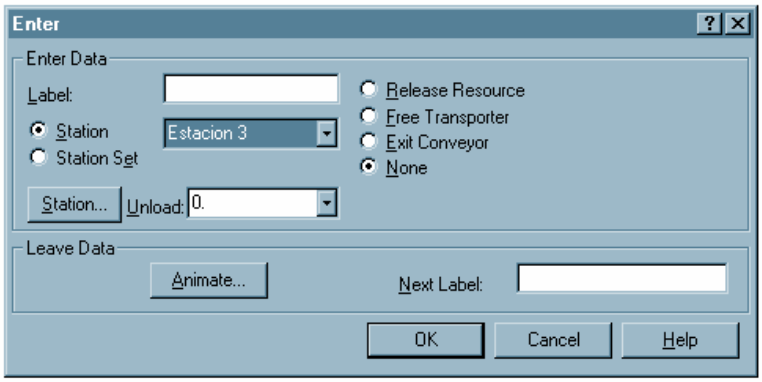


La Estación 3 no puede modelarse de esta forma puesto que las dos máquinas son distintas. Si fueran iguales se podría poner un módulo *Server* con *Capacity* igual a 2 pero esto no es posible en nuestra situación. En lugar de esto se va a utilizar el conjunto de módulos *Enter-Process-Leave*.

Módulos Enter-Process-Leave.

Estos módulos simulan las distintas componentes de un módulo *Server* pero, al implementarlas separadamente, permiten una mayor flexibilidad y mayores posibilidades de simular estaciones complejas. Un ejemplo de esto es la Estación 3 que, al disponer de dos recursos de distinta capacidad, no podía ser modelado con un módulo *Server*.

El módulo *Enter* es muy similar a la sección *Enter Data* del módulo *Server* con la salvedad de que se puede entrar en una estación o en un *Set* de estaciones.

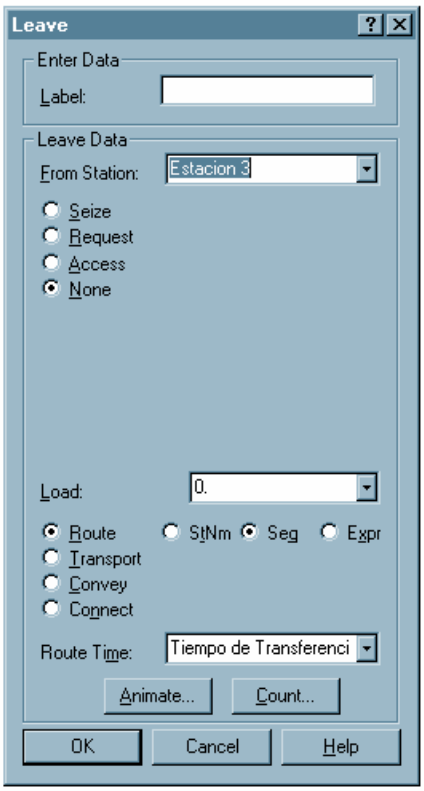


En el módulo *Process* se va a considerar que tiene el *Set* de *Resources* ya definido anteriormente, Estaciones 3, cuyos componentes, Nuevo 3 y Viejo 3, se irán ocupando de forma cíclica.

Se define un “Indice” que devolverá la máquina de la estación a la que corresponde procesar la pieza. El tiempo de proceso que se tendrá será el definido en la secuencia multiplicado por el Factor creado en el módulo *Variables* indexado por este Indice.

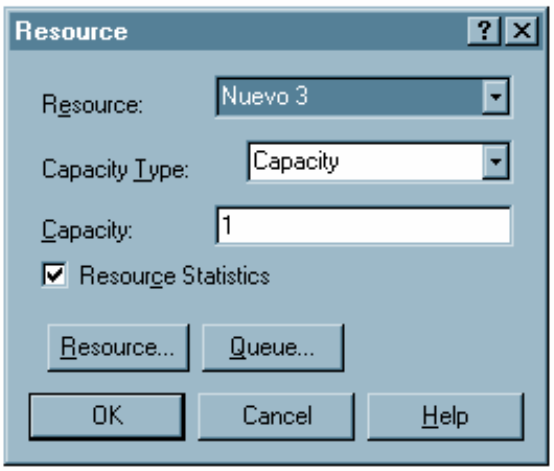
```
Tiempo de Proceso*Factor( Indice )
```

El módulo *Leave* es similar a la sección *Leave Data* del módulo *Server*. Para este caso se define de la siguiente forma:



Módulos Resource.

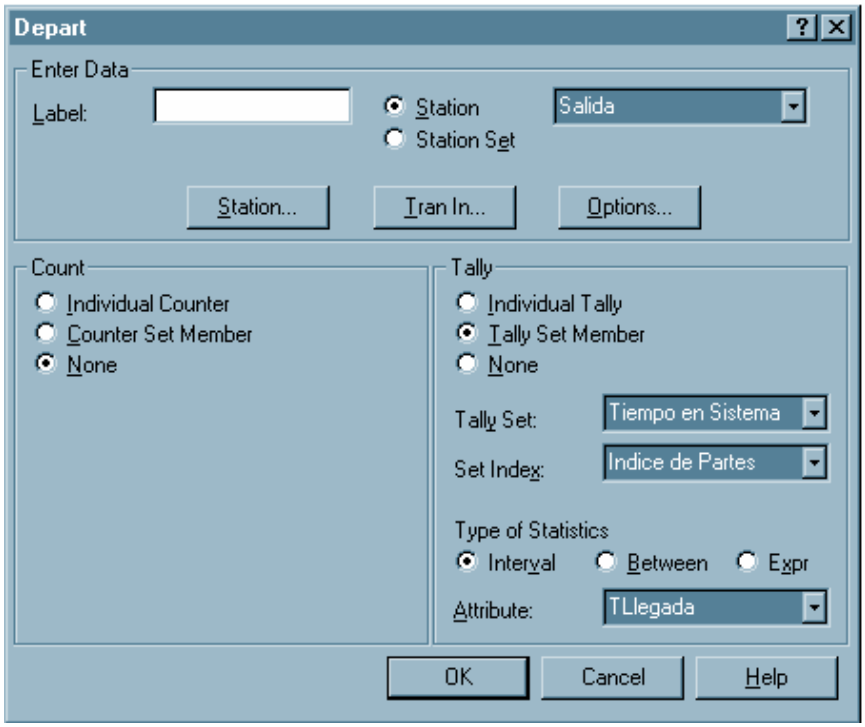
Un detalle que quedaba pendiente en el apartado anterior era que no se habían definido explícitamente los recursos con que contaba la Estación 3. Se habían referenciado en el *Resource Set* pero no se habían definido realmente. Esto se hace con el módulo *Resource.*



Módulo Depart.

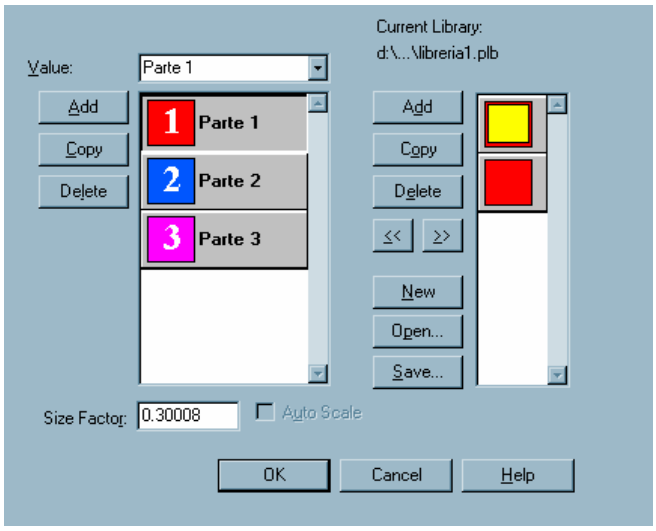
Es el módulo de salida del sistema. A él van todas las partes que terminan la secuencia que les ha sido asignada.

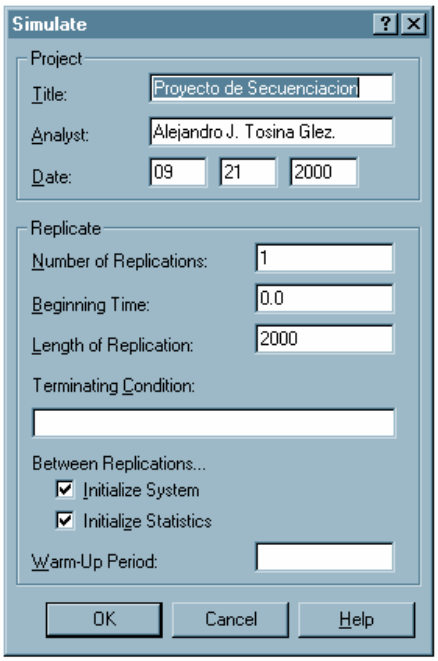
Lo único especial que se incluye en este módulo es marcar la casilla *Tally Set Member* para tener un *Set* de *Tallies* indexado por el Indice de Partes para poder medir el tiempo de permanencia en el sistema de cada parte por separado.



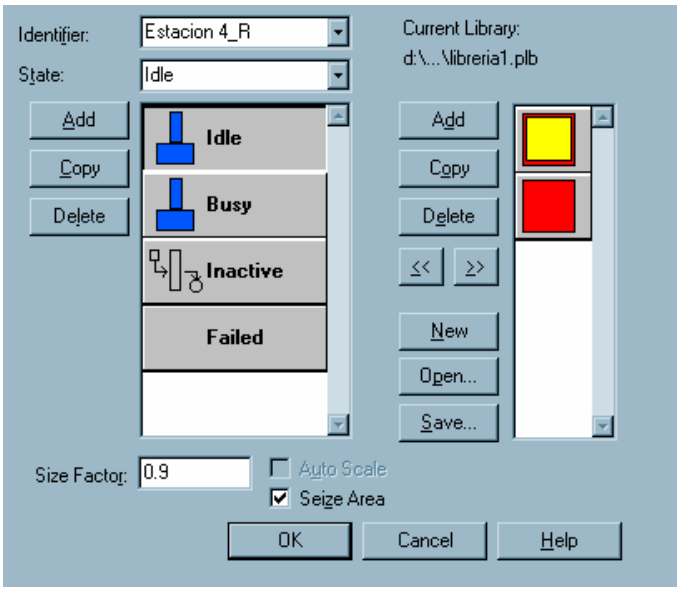
Módulo Simulation.

En este módulo se va a incluir el tiempo de duración de la simulación y los dibujos que van a representar a las distintas partes en la simulación.



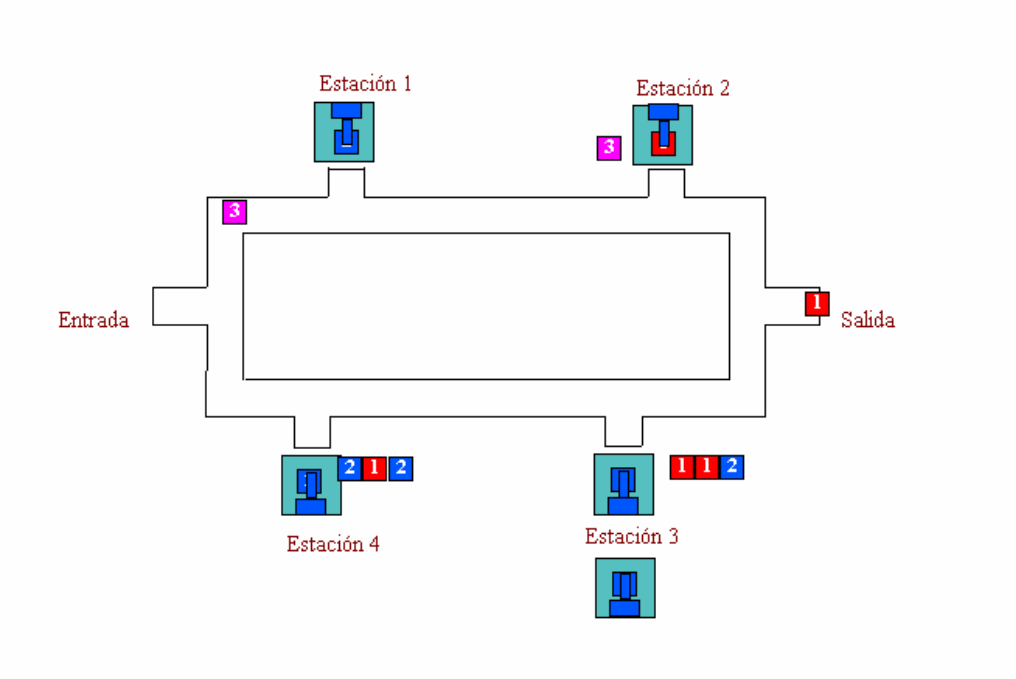


Los dibujos que representan a los servidores también se han cambiado. Esto se hace pinchando en el dibujo que viene por defecto.



Ahora ya solo queda ejecutar la simulación y observar los resultados.

Simulación del sistema.



Los resultados obtenidos son los siguientes:

ARENA Simulation Results
Alejandro Javier Tosina Glez. - License #9400000

Summary for Replication 1 of 1

Project: Proyecto de Secu
Analyst: Alejandro J.Tosi

Run execution date : 9/30/2000
Model revision date: 9/21/2000

Replication ended at time : 2000.0

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
TSistema Parte 1	133.16	(Insuf)	55.626	244.72	37
TSistema Parte 2	173.23	(Insuf)	91.773	306.74	68
Estaciones 3_Q Queue T	14.781	(Insuf)	.00000	56.120	142
TSistema Parte 3	97.596	(Insuf)	46.333	196.32	35
Estacion 4_R_Q Queue T	29.963	(Insuf)	.00000	71.920	110
Estacion 1_R_Q Queue T	18.339	(Insuf)	.00000	85.057	152
Estacion 2_R_Q Queue T	15.013	(Insuf)	.00000	49.152	223

DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
# in Cell 3 Machines_Q	.00000	(Insuf)	.00000	.00000	.00000
Viejo 3 Busy	.87330	(Insuf)	.00000	1.0000	1.0000
Estacion 4_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
# in Estaciones 3_Q	1.0604	(Insuf)	.00000	5.0000	2.0000
Nuevo 3 Busy	.83712	(Insuf)	.00000	1.0000	1.0000
Viejo 3 Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Estacion 2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Estacion 1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
# in Estacion 4_R_Q	1.6872	(Insuf)	.00000	6.0000	3.0000
Estacion 4_R Busy	.87733	(Insuf)	.00000	1.0000	1.0000
# in Estacion 2_R_Q	1.6845	(Corr)	.00000	7.0000	2.0000
Estacion 2_R Busy	.85679	(Insuf)	.00000	1.0000	1.0000
# in Estacion 1_R_Q	1.4721	(Insuf)	.00000	10.000	4.0000
Estacion 1_R Busy	.84052	(Insuf)	.00000	1.0000	1.0000
Nuevo 3 Available	1.0000	(Insuf)	1.0000	1.0000	1.0000

Simulation run time: 2.92 minutes.
Simulation run complete.

La simulación de este sistema para diversas decisiones de planificación, es decir, para distintas ordenaciones de las tareas a realizar para cada parte, proporciona una potente herramienta para simular y optimizar sistemas complejos.

Simulación de Sistemas de Producción.

Introducción a los Sistemas de Producción.

La herramienta de simulación que está siendo estudiada puede ser aplicada en múltiples campos y actividades. Cuando se profundiza en la comprensión de un sistema se pueden definir sus características cada vez con mayor precisión. Todos los sistemas de producción presentan muchas características similares entre ellos aunque puedan variar en más o menos grado de detalle. Debido a esta similitud a grandes rasgos que presentan estos sistemas se pueden desarrollar modelos que sean aplicados a varios sistemas de producción distintos, con pequeñas modificaciones en su estructura.

Todos los sistemas de producción combinan la existencia de productos, materias primas y productos manufacturados en mayor o menor grado, con los operadores y las herramientas necesarias para realizar el trabajo. De esta forma los sistemas de producción tendrán componentes similares. La diferencia entre ellos estará en las distintas formas en las que se pueden combinar todos estos elementos para formar el sistema. Esta complejidad a la hora de ordenar los componentes de los sistemas de producción es la que permite que la simulación sea una herramienta tan útil en su planificación y en el análisis de su comportamiento para poder garantizar su correcto funcionamiento.

La simulación puede ser aplicada en muchos aspectos de los sistemas de producción. Dos áreas son de un interés particular a la hora de estudiar este tipo de sistemas:

- *Job-shops*: Simulación de distintas reglas de despacho a la hora de estudiar la eficiencia en el empleo de las distintas herramientas.
- *Flow-shops*: Estudio del efecto de fallos en los componentes del sistema.

Sistemas de Fabricación Flexible (FMS).

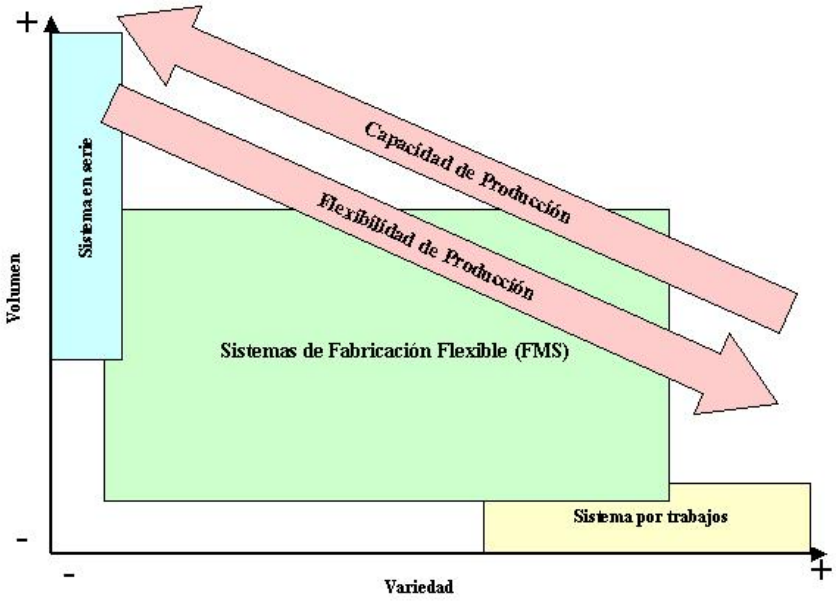
Dentro del amplio conjunto de sistemas de producción que existen en la realidad será objeto de estudio en particular los sistemas de fabricación flexible (**FMS:** *Flexible Manufacturing Systems*). Estos sistemas están compuestos de dos o más máquinas de control numérico interconectadas con equipos de trabajo y supervisión completamente automatizados y dependiendo todos de un ordenador central que se encarga de regular las secuencias de tareas que cada elemento del sistema debe realizar.

Como se comentó anteriormente existen dos formas básicas de trabajar a la hora de planificar un sistema de producción:

- *Flow-shop*: Taller serial. Estos son sistemas donde todos los productos pasan por las mismas estaciones, donde se realizan las mismas operaciones. Un ejemplo sencillo sería una cadena de montaje en una fábrica de coches. En estos casos **la producción es muy elevada** pero **la versatilidad es reducida** por lo que se puede producir un número muy reducido de productos.
- *Job-shop*: Taller por trabajos. Aquí se tienen máquinas independientes, formando células de producción que realizan una tarea concreta. Cada producto puede seguir una secuencia concreta entre las distintas máquinas que componen el sistema de producción. Un ejemplo podría ser un taller de reparación de vehículos, con distintos instrumentos. Un determinado vehículo sólo necesitará un número determinado de reparaciones distintas y en un orden que será determinado en función de las necesidades del taller, sin perjuicio del objetivo final de reparación. Con este tipo de sistema se puede obtener una **variedad mayor** de productos que con el *Flow-shop*, pero a costa de producir una **cantidad menor** de productos finales.

Los sistemas de fabricación flexible intentan conjugar los dos objetivos deseables de conseguir una producción muy elevada pero sin renunciar a la versatilidad y a la variedad de productos que puedan satisfacer el mercado.

Estos argumentos se pueden observar de una manera más clara en el siguiente gráfico, en el que se representa la relación entre el volumen de producción y la variedad de productos en los sistemas de fabricación:



Los sistemas de fabricación flexible consiguen este objetivo de conjugar la versatilidad de la producción con la deseada fabricación de una gran cantidad de productos mediante la elevada automatización y el riguroso control de sus componentes, ya sea mediante el uso de robots, cintas transportadoras o vehículos de guiado automático (**AGV**: Automated Guided Vehicles). Estos elementos consiguen un alto rendimiento en el proceso productivo y forman una parte esencial de estos sistemas de producción.

Para observar el funcionamiento de este tipo de sistemas se estudiará una planta de producción que combinará este tipo de técnicas a fin de que se pueda comprobar por simulación su eficiencia y operatividad.

Simulación de un Sistema de Producción.

El sistema que va a ser modelado trabajará con dos tipos de entidades, Tipo1 y Tipo2, que van a seguir rutas diferentes entre los distintos procesos que conforman la planta de producción.

Una vez que las entidades, piezas a procesar, llegan al sistema, se encaminan a través de una cinta transportadora hasta uno de los dos módulos de selección, donde se separan según el tipo y se encaminan hacia el proceso que les corresponda a continuación.

Cuando salen de los módulos de selección, las entidades se mueven entre los distintos procesos mediante el uso de vehículos de guiado automático (AGV). Este movimiento se puede corresponder con dos arquitecturas de funcionamiento distintas de la planta, las cuales serán estudiadas y comparadas para ver cuál es la mejor. Estas dos arquitecturas son las siguientes:

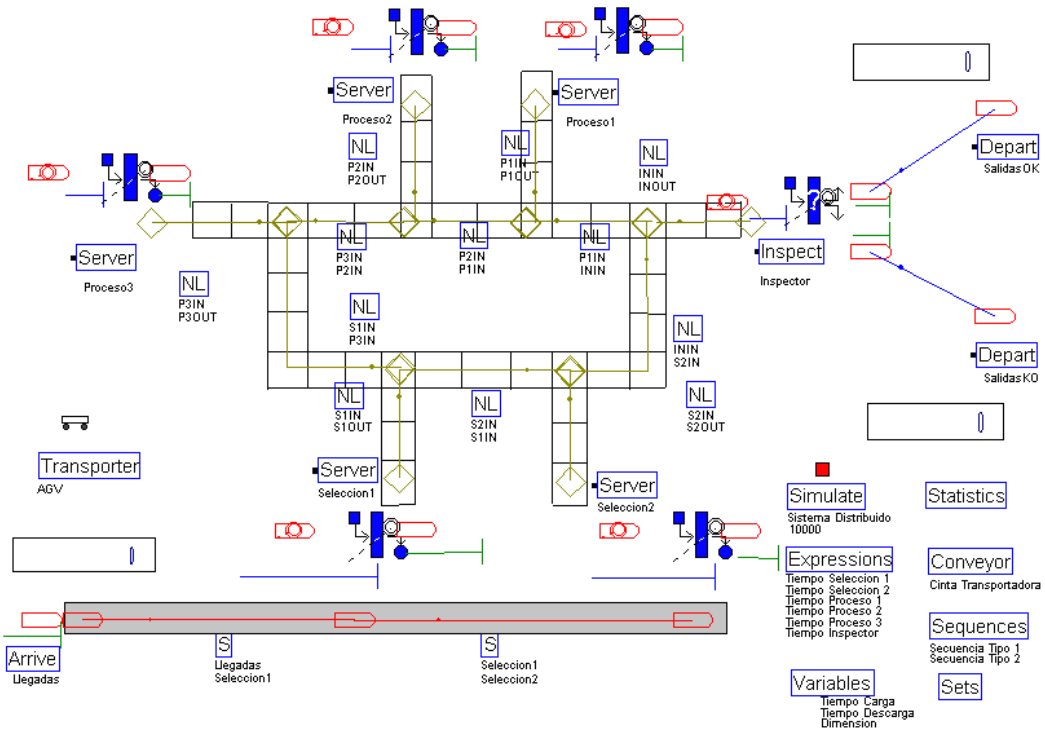
- **Sistema de almacén distribuido**: No se dispone de almacén central. Cada proceso por separado tiene una pequeña cola donde esperan las entidades para ser procesadas. Los movimientos de los AGV serán de proceso a proceso.
- **Sistema de almacén centralizado**: Se tiene un almacén central al que irán todas las entidades después de cada proceso. Los movimientos de los vehículos serán siempre entre algún proceso, de producción o selección, y dicho almacén central.

Después de ser procesadas según su secuencia correspondiente, las entidades pasan a un módulo de inspección, donde se aceptarán o se rechazarán en función a la correcta producción de la pieza o no. Si no se acepta, la pieza se considera perdida y no hay capacidad de recuperación o reutilización.

Sistema de Almacén Distribuido.

Modelo.

El modelo de este sistema es el siguiente. Hay que tener en cuenta que para evitar colisiones entre los vehículos automáticos, éstos van a seguir una ruta circular que los irá llevando por los distintos nodos de acceso a las estaciones.

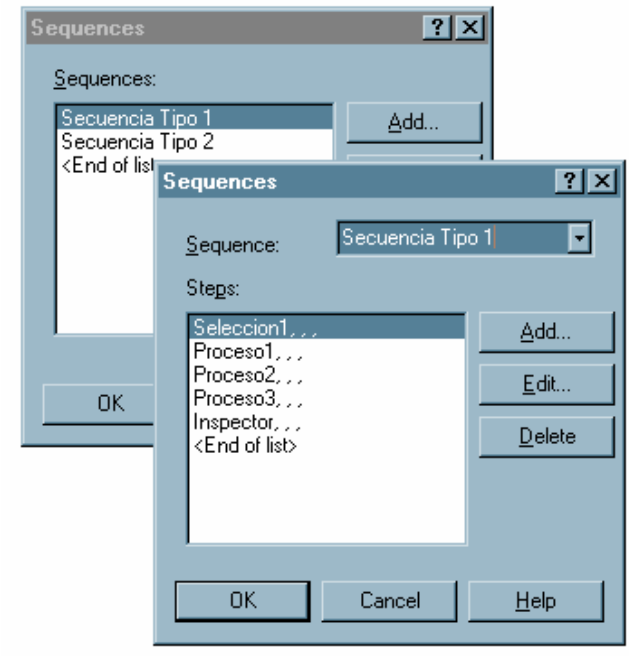


Módulo Sequences.

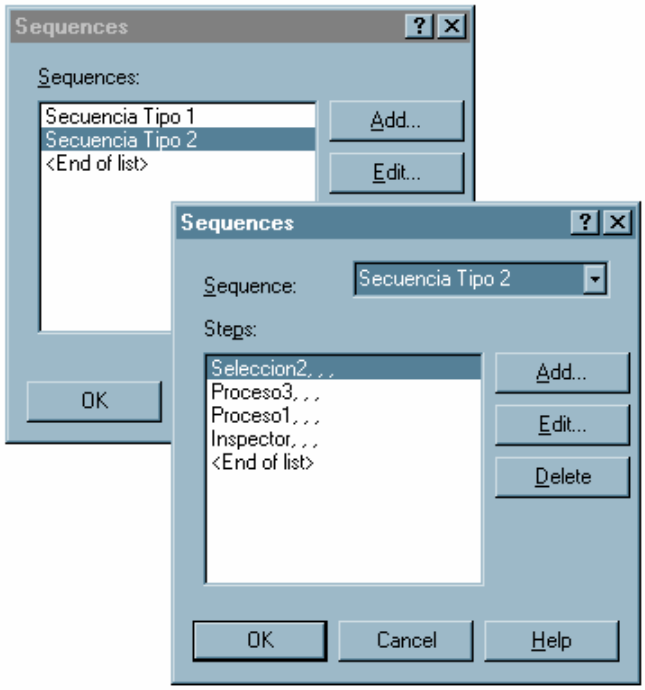
Este módulo indica las secuencias de procesos que van a seguir las entidades de ambos tipos que lleguen al sistema. Estas secuencias representan las listas de servidores que van a requerir para su proceso completo en el sistema.

Cada tipo de entidad seguirá una secuencia diferente debido a sus características distintivas y al producto final que se quiera obtener de ella.

Las entidades de tipo 1 siguen la siguiente secuencia.



Las entidades de tipo 2 siguen otra distinta.

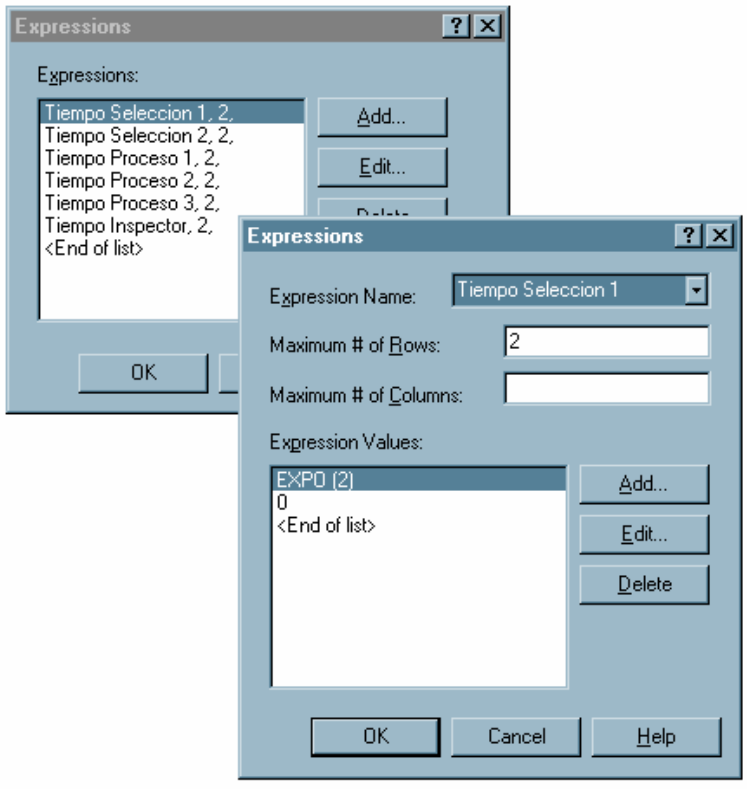


Módulo Expressions.

Este módulo se encarga de definir expresiones asociadas a un nombre en concreto. Cuando se hace referencia a ese nombre en el modelo se evalúa su expresión asociada y se sustituye el nombre por su valor.

En el presente modelo se van a definir expresiones para los tiempos que van a tardar las entidades en ser procesadas por las distintas máquinas. Estos tiempos son los siguientes.

Tipo de Pieza	Estación Tiempo	Estación Tiempo	Estación Tiempo	Estación Tiempo	Estación Tiempo
1	Selección 1 EXPO(2)	Proceso 1 EXPO(10)	Proceso 2 EXPO(8)	Proceso 3 TRIA (6,8,10)	Inspector EXPO(2)
2	Selección 2 EXPO(4)	Proceso 3 EXPO(10)	Proceso 1 EXPO(12)	Inspector EXPO(3)	



Módulo Variables.

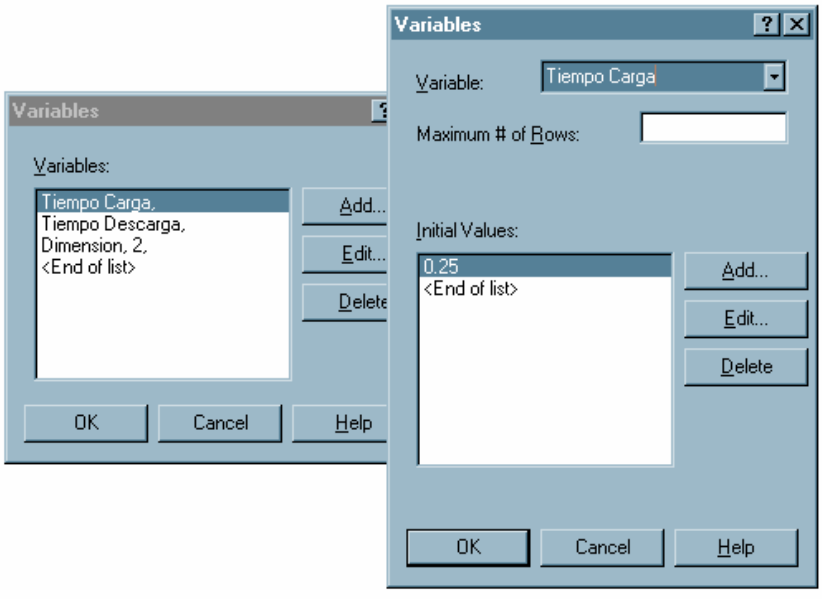
Este módulo define un valor que se le asocia a un nombre. Se diferencia del módulo Expressions en que asocia un valor en concreto, un número, y no una expresión que deba ser evaluada previamente.

Las variables que se van a definir son las siguientes.

Variable	Bidimensional	Valor
Tiempo Carga	NO	0.25
Tiempo Descarga	NO	0.25
Dimensión	SI	(1,2)

El Tiempo Carga y Tiempo Descarga se refieren a los tiempos que tardan las entidades en cargarse o descargarse de la cinta transportadora y los AGV. Se supone, por simplicidad, que tardan lo mismo ambos tipos de entidad en ambos tipos de aparato.

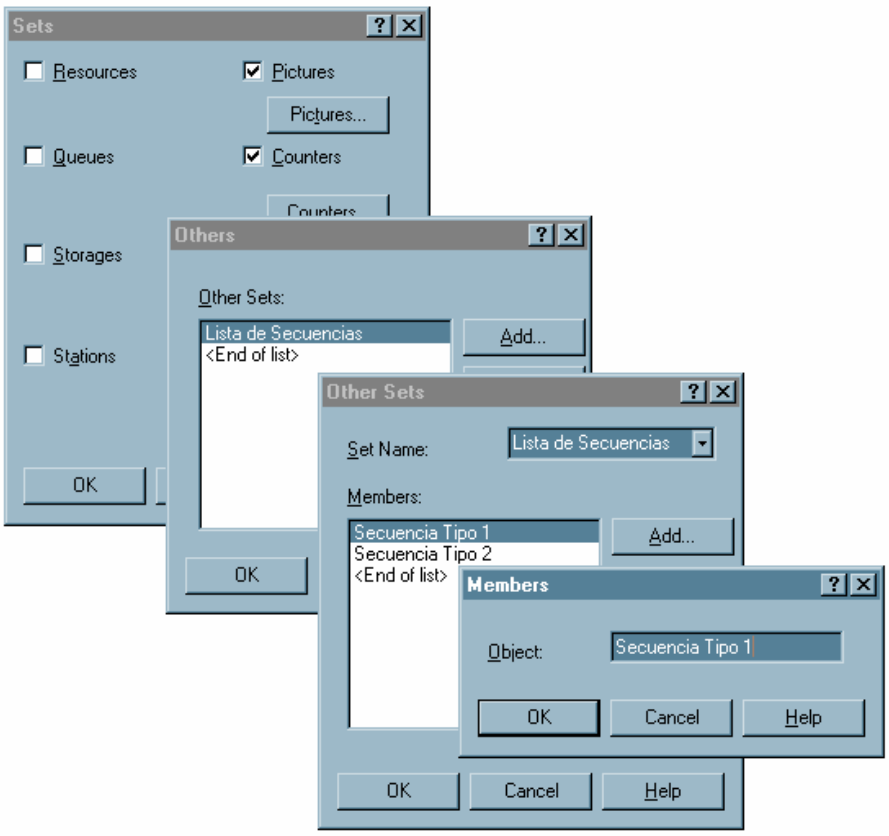
Dimensión es una variable que indica cuántas celdas de la cinta transportadora ocupa una entidad. Este concepto se explicará más adelante. Sólo se resalta aquí que las piezas de tipo 1 ocupan una celda mientras que las de tipo 2 ocupan 2 celdas.



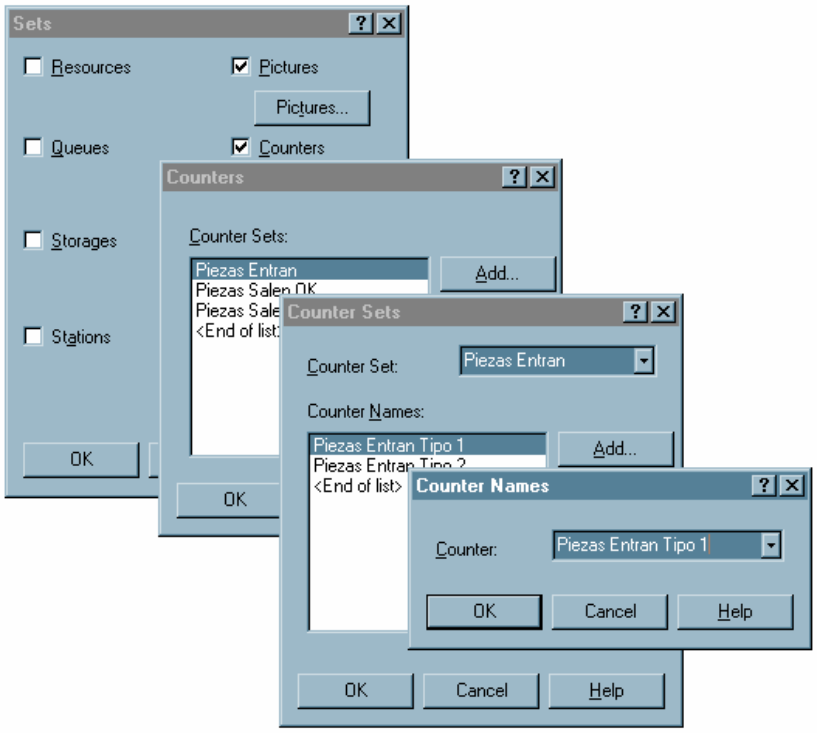
Módulo Sets.

Este módulo sirve para definir conjuntos de elementos similares. Será necesario para definir conjuntos de *Sequences*, *Counters* y *Tallies*, ya que, además de las secuencias de procesos, hará falta medir estadísticos de las entidades diferenciándolas según su tipo.

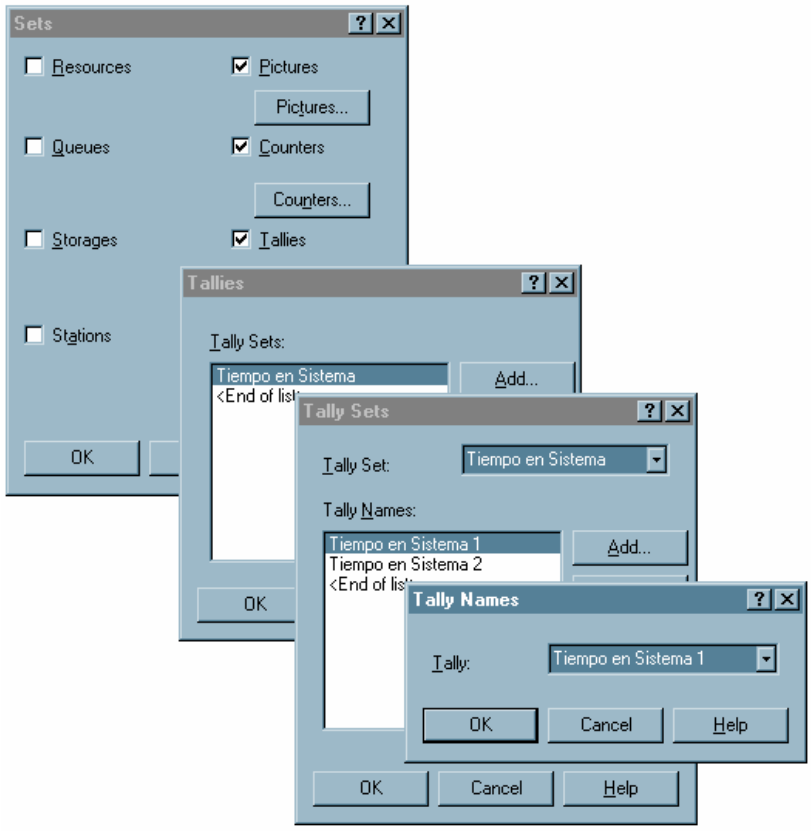
El Set de Sequences es el siguiente.



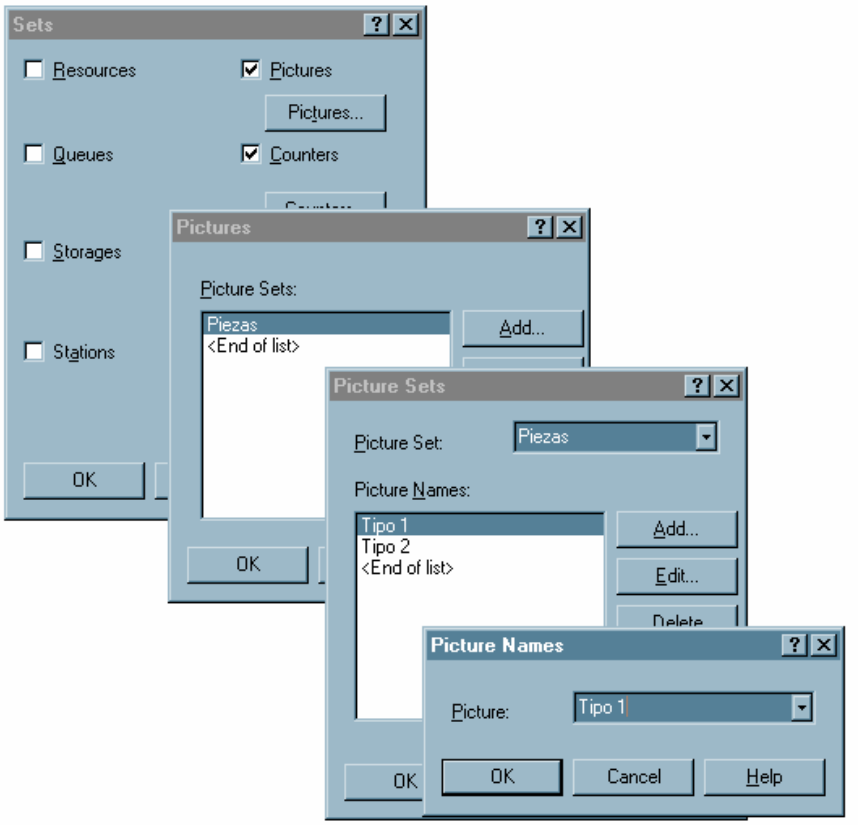
El Set de Counters es el siguiente.



El Set de Tallies es el siguiente.



Un último Set que hay que definir es el Set de Pictures. Se tendrá una imagen distinta por cada tipo de pieza que esté definida.



Módulos de Transferencia.

Estos módulos hacen referencia a los medios que utilizan las entidades para moverse por el sistema. Hasta ahora se han visto únicamente los ***Routes***, que no implican más que las entidades pasan de una estación a otra por sus propios medios y sin problemas de capacidad en el sistema para que se muevan. Los retardos que se producían en los desplazamientos de las entidades entre los módulos que representaban a las estaciones no podían reflejar en ningún caso posibles problemas de congestión en el sistema. Para reflejar este tipo de casos existen otros elementos de simulación en Arena que se describen a continuación.

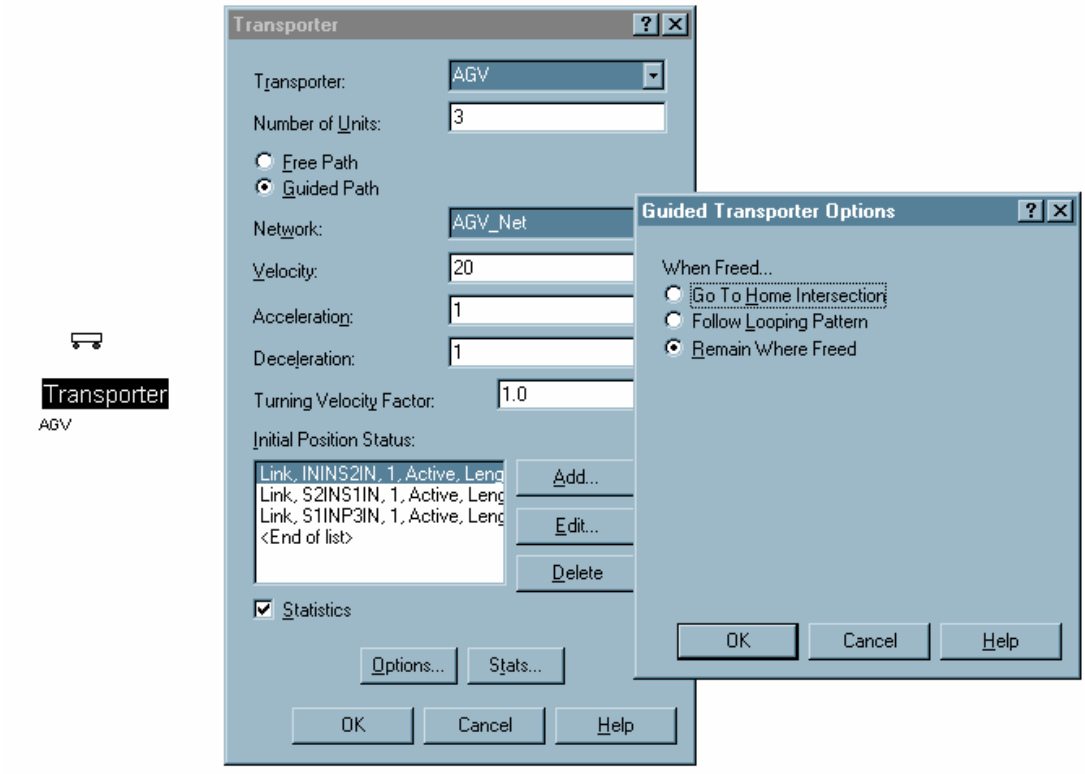
Otra forma de transferir entidades entre módulos del sistema es mediante los ***Resource-Constrained***. Se trata de recursos con capacidad limitada que simulan enlaces de comunicación entre elementos. Estos recursos deben ser reservados por la entidad que quiera ser transportada y liberados una vez que dicha entidad llegue a su destino. Esta forma de transporte no será utilizada en este sistema de producción y parece más indicada para la simulación de redes de comunicaciones.

Los vehículos de guiado automático (AGV) que funcionan en el sistema que va a ser estudiado tienen un elemento en particular que los simula dentro de Arena. Este elemento es el ***Transporter***. Los *Transporter* son llamados para coger una entidad, la cargan, la transportan y luego son descargados en el destino, con lo que quedan libres para ser utilizados por más entidades. Estos transportadores pueden ser *Free-Path*, se pueden mover libremente por el sistema sin más retardos que el producido por su propia velocidad, o *Guided-Path*, en cuyo caso su movimiento está definido por una determinada red.

En el sistema de producción a estudiar se utiliza una red circular de enlaces monodireccionales que recorre una serie de nodos de acceso a enlaces *Spur* que conectan la red con las estaciones para producir la carga y descarga de las entidades. Estos enlaces *Spur* pueden ser únicamente utilizados por un solo transportador, de forma que no se producirán colisiones entre los distintos AGV.

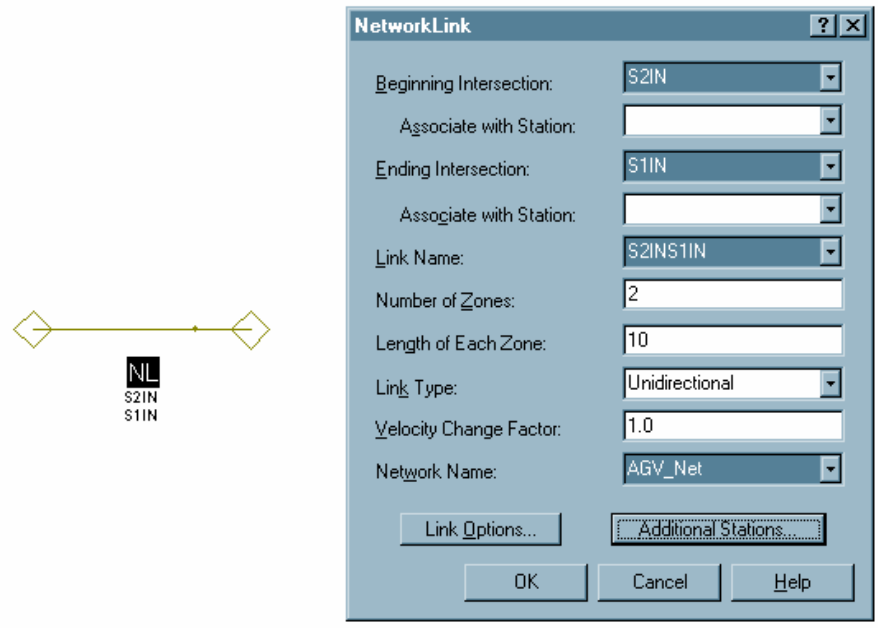
En el presente sistema de producción se van a tener tres *Transporters* que se moverán cíclicamente siguiendo la dirección de las agujas del reloj y, en cuanto se les solicite, se desviarán hacia algún enlace *Spur* para cargar alguna entidad. Posteriormente seguirán viajando por la red circular hasta llegar a la estación destino y descargar la pieza. Una vez liberados esperarán a que otra entidad les requiera para su transporte.

Los transportes de este modelo se definen de la siguiente forma.

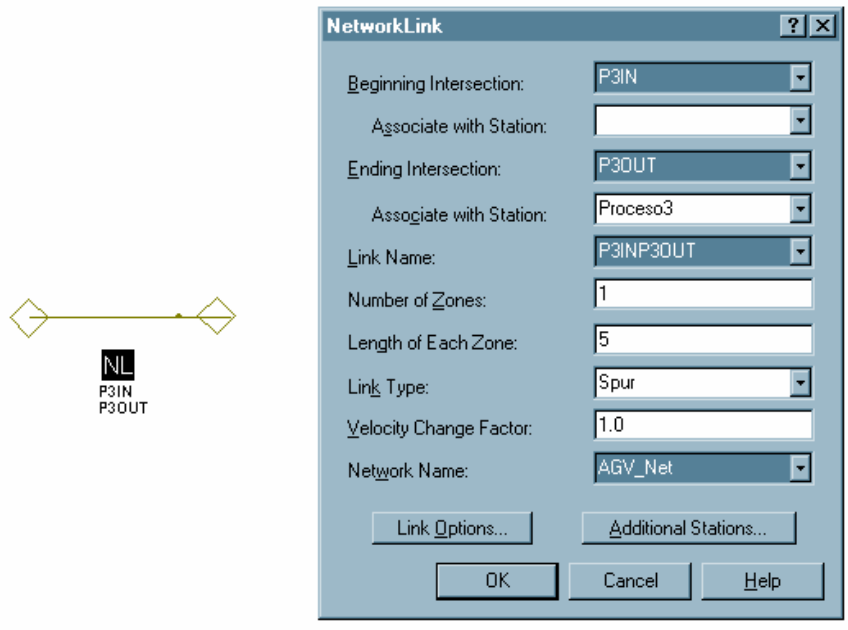


Los enlaces de la *Network* que seguirán los AGV tienen las siguientes características.

- Enlaces monodireccionales (red circular de transporte).



- Enlaces Spur (red de acceso).



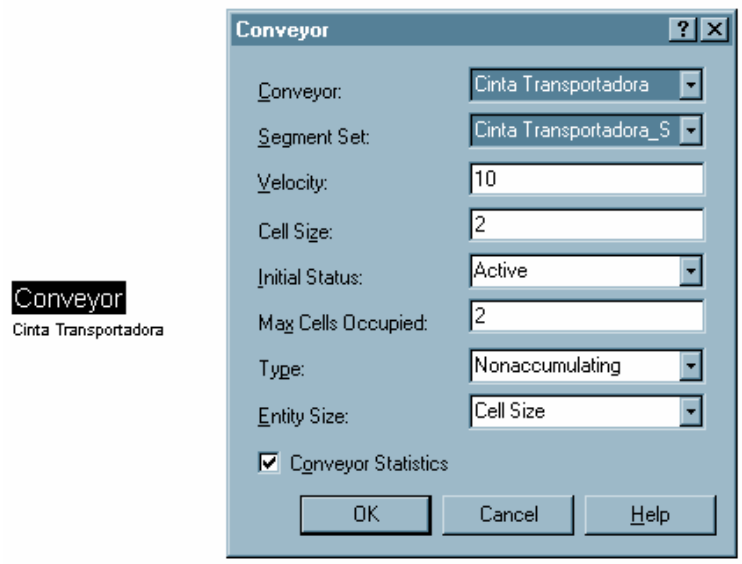
Para simular el funcionamiento de la cinta transportadora,, Arena dispone de un elemento creado para tal fin, el ***Conveyor***. El *Conveyor* consiste en celdas de igual longitud que se están moviendo continuamente en una dirección determinada. La entidad que quiera ser transportada, que ocupará un número determinado de celdas, debe esperar a que halla suficiente espacio libre en el *Conveyor* para poder acceder a él. Por este motivo es importante definir bien el tamaño de la celda, pues una celda pequeña implica un mayor aprovechamiento del espacio pero lleva implícito una mayor carga de cálculo, ya que su número será muy grande, mientras que una celda grande aumenta la eficiencia computacional del sistema pero se pierden prestaciones en cuanto a granularidad y espacio, puesto que una entidad que quiera acceder al *Conveyor* cuando una celda acaba justo de pasar, tendrá que esperar a que termine de pasar la siguiente, que será grande, para poder acceder.

Existen dos tipos básicos de *Conveyors*:

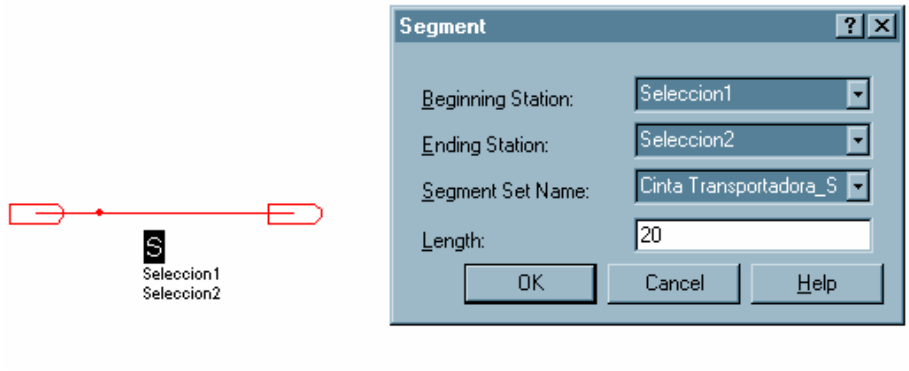
- *Accumulating*: Cuando una entidad intenta acceder a un *Conveyor*, éste no se para, con lo cual las entidades que ya estaban en él se van acumulando detrás de la nueva esperando a que ésta comience a desplazarse.
- *Nonaccumulating*: El *Conveyor* se detiene mientras una entidad está accediendo a él. De esta forma el espacio entre entidades dentro de la cinta no cambia, no produciéndose el efecto acumulativo.

En este sistema, la cinta transportadora constará de dos segmentos de 20 metros cada uno; uno entre el módulo de llegada y el puesto de Selección 1 y otro entre Selección 1 y Selección 2. El tipo de *Conveyor* se va a definir como *Nonaccumulating*. La velocidad de la cinta será de cuatro metros por segundo y el tamaño de la celda básica será de dos metros. Las entidades, como se define en la variable Dimensión, ocuparán una celda (dos metros) las de Tipo 1, y dos celdas (cuatro metros) las de Tipo 2.

La cinta transportadora se define a continuación.



Los segmentos de la cinta transportadora se definen de la siguiente forma.

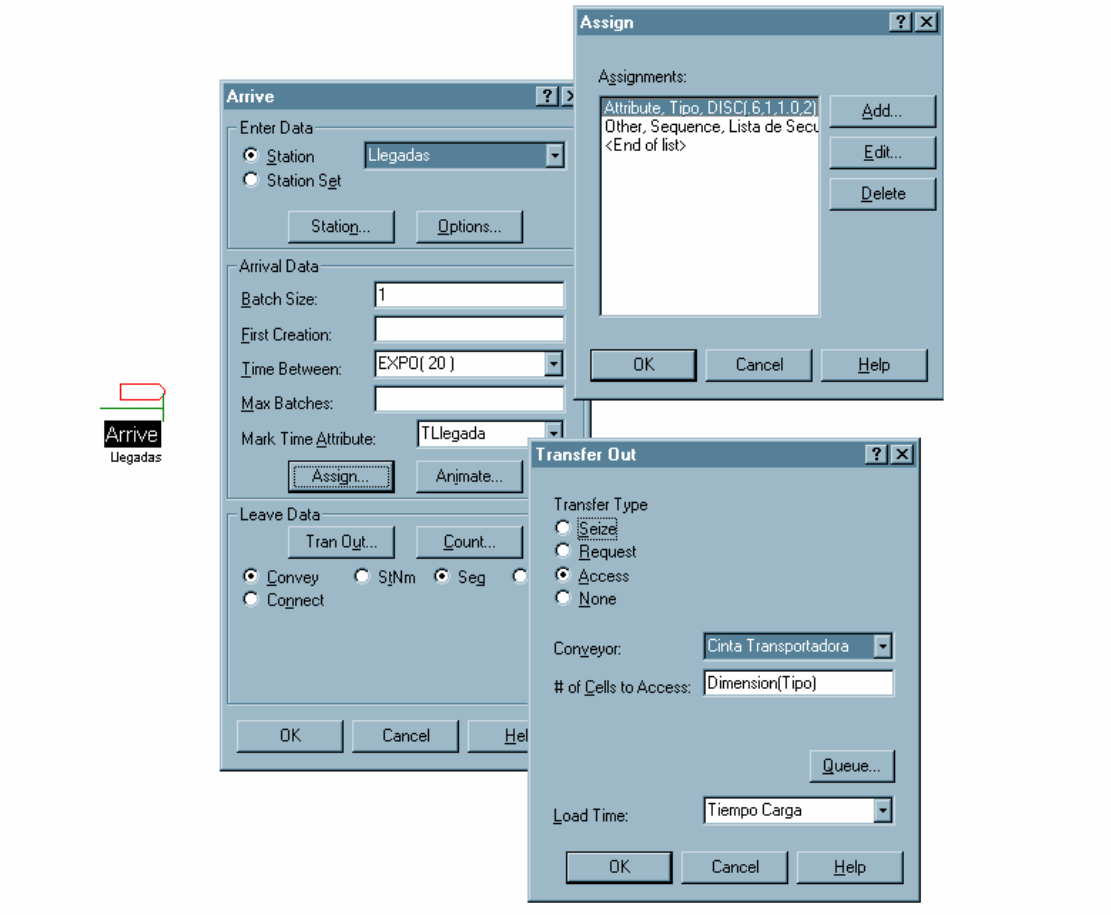


Módulo de Llegadas.

El módulo *Arrive* define cómo van llegando las entidades al sistema y les asigna algunos atributos.

Las entidades llegarán al sistema siguiendo una distribución exponencial de media 20 minutos. El 60% de las mismas será de Tipo 1 (piezas rojas) mientras que el 40% restante será de Tipo 2 (piezas azules).

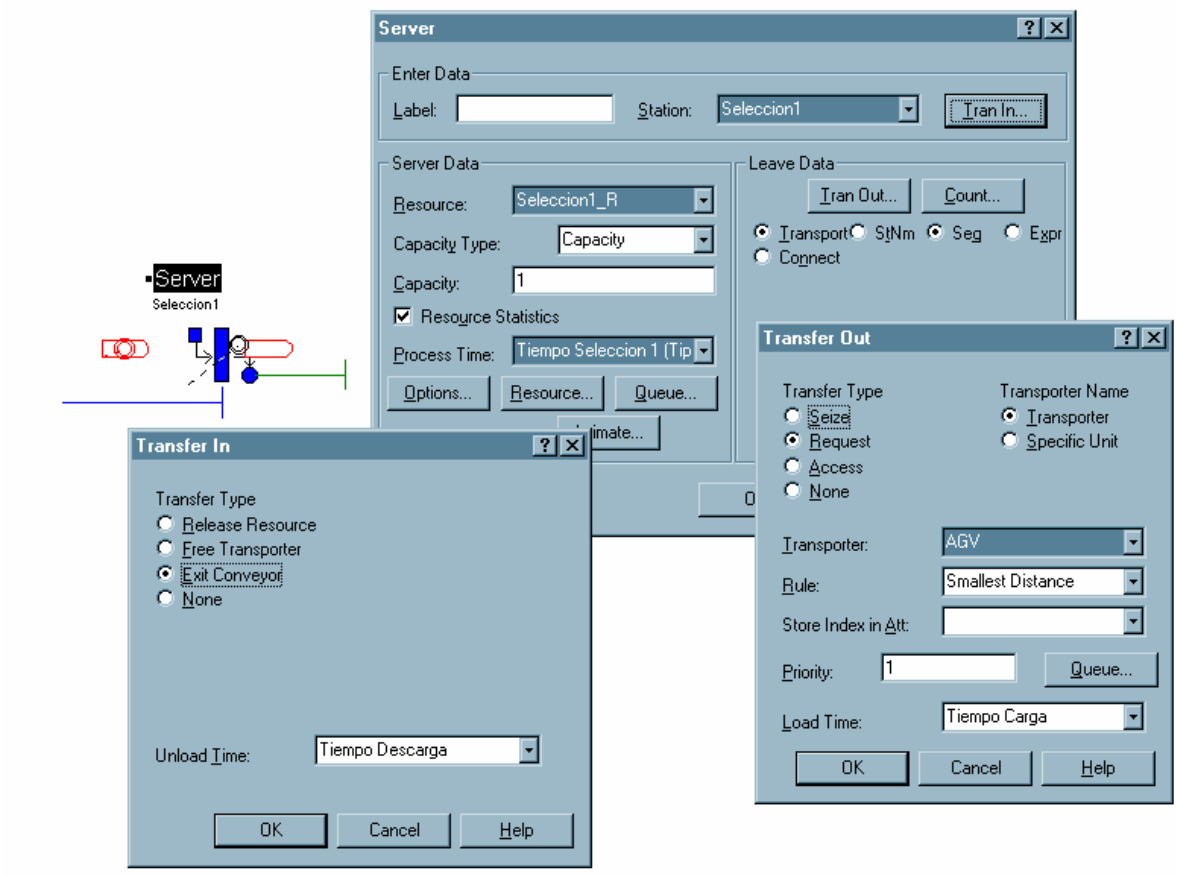
A continuación pasarán a ser procesadas siguiendo su secuencia de operaciones correspondiente, para lo cual serán separadas según su tipo en los módulos de Selección 1 y Selección 2. Para llegar a estos módulos de selección deben acceder primero a la cinta transportadora que las dirigirá hacia ellos.



Módulos de Selección.

En estos módulos se cogen las entidades del tipo que tengan definido según la secuencia de operaciones y se envían a las estaciones de proceso pertinentes.

Para recoger las entidades de la cinta transportadora necesitan un Tiempo de Descarga de 0.25 minutos. Posteriormente preparan dichas entidades para ser transportadas por los AGV. Este proceso les lleva un Tiempo de Selección definido por el módulo *Expressions*. A continuación solicitan el uso de un AGV para transportar la entidad a los procesos correspondientes. Una vez que el vehículo ha llegado, tardan un Tiempo de Carga de 0.25 minutos en disponer la pieza para ser transportada, tras lo cual quedan libres para preparar otra entidad.

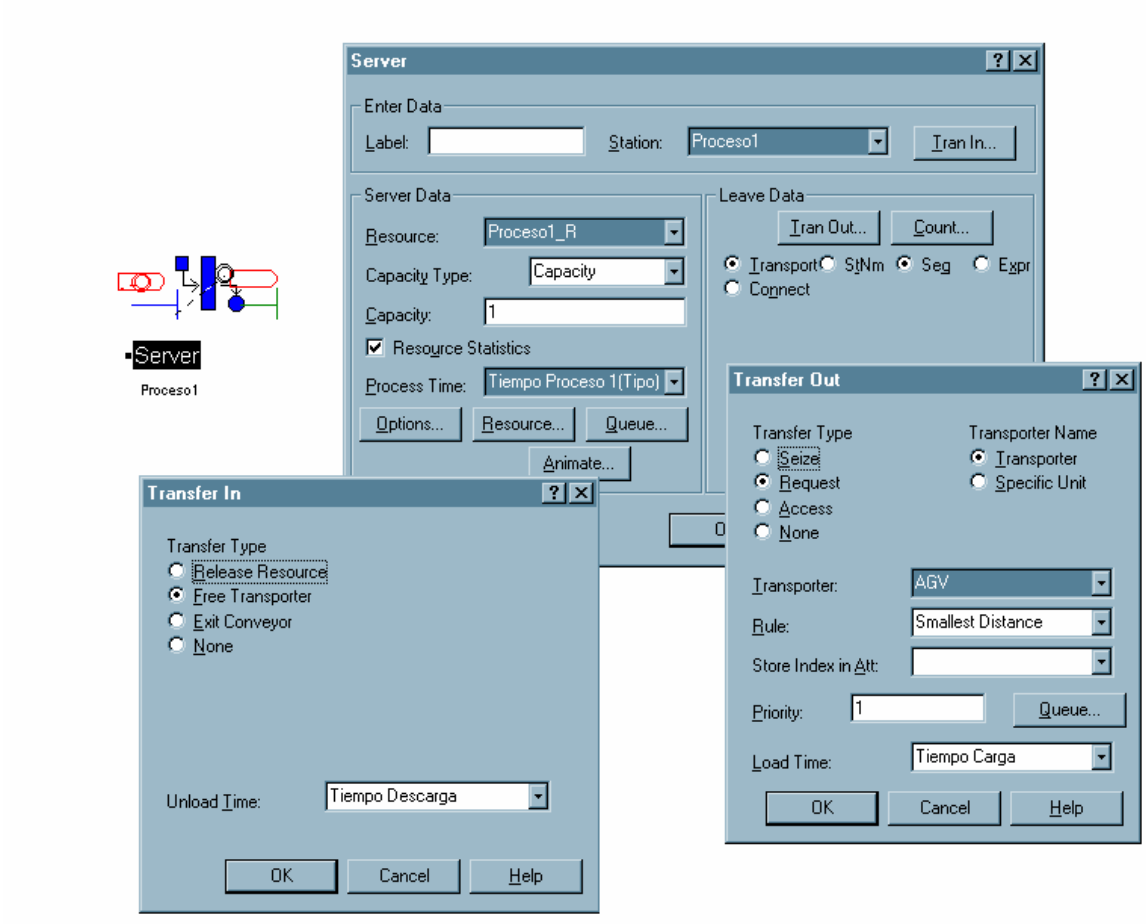


Módulos de Proceso.

Estos son los módulos que procesarán las entidades para obtener los productos finales del sistema.

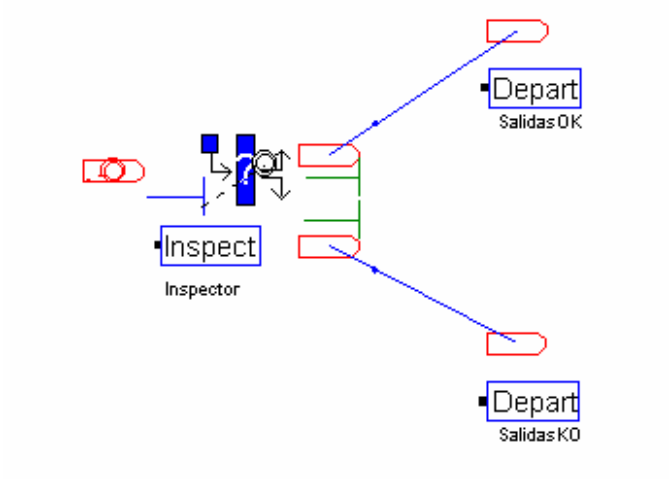
Las entidades se moverán entre ellos siguiendo las secuencias correspondientes utilizando la red de vehículos de guiado autónomo.

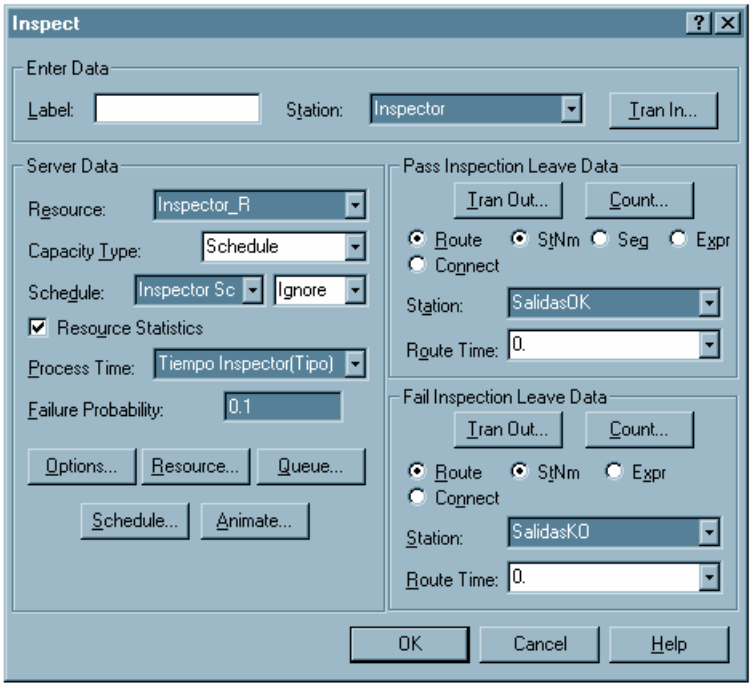
Estos módulos son estructuralmente iguales y se describen a continuación.



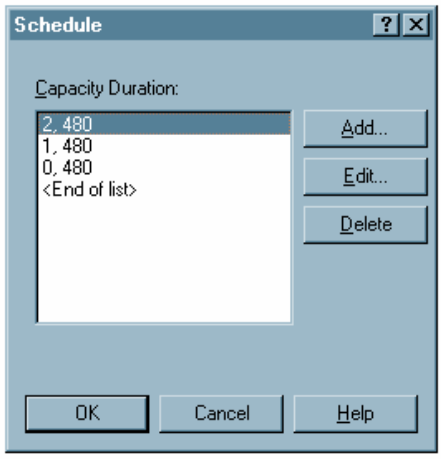
Módulo Inspector.

Este módulo se encarga de discriminar las entidades que han salido correctamente de la cadena de proceso, de las que han salido defectuosas. La probabilidad de fallo es del 1%.

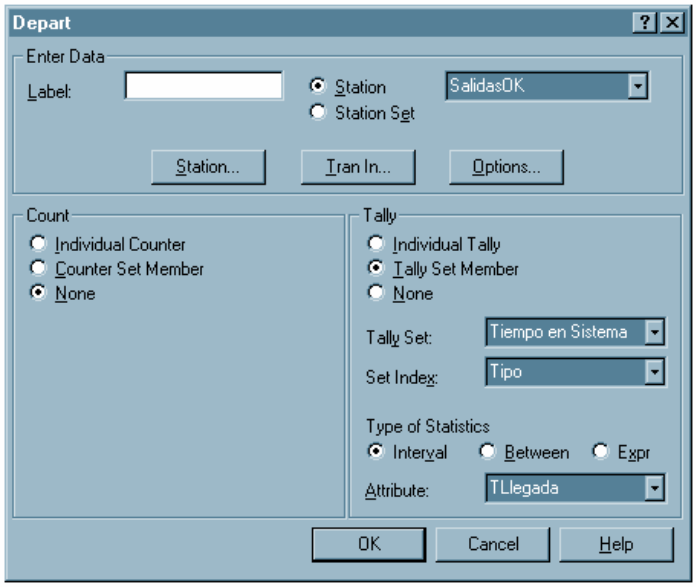




Hay que tener en cuenta que la supervisión se realiza por turnos de 8 horas (480 minutos). En el turno de mañana hay dos personas revisando las piezas, en el de tarde solo hay una, mientras que de noche nadie supervisa las piezas que van terminando de ser procesadas. Éstas se van almacenando en la cola de entrada del servidor hasta la mañana siguiente, momento en el que esta estación volverá a disponer de personal (recursos) para revisar las piezas. Esto se modela con la opción *Schedule* de la estación.



La salida de esta estación Inspector se realiza hacia los módulos *Depart* Salidas OK y Salidas KO (según el resultado de la inspección) mediante *Routes* de duración 0. En los módulos *Depart* las entidades abandonan definitivamente el sistema.

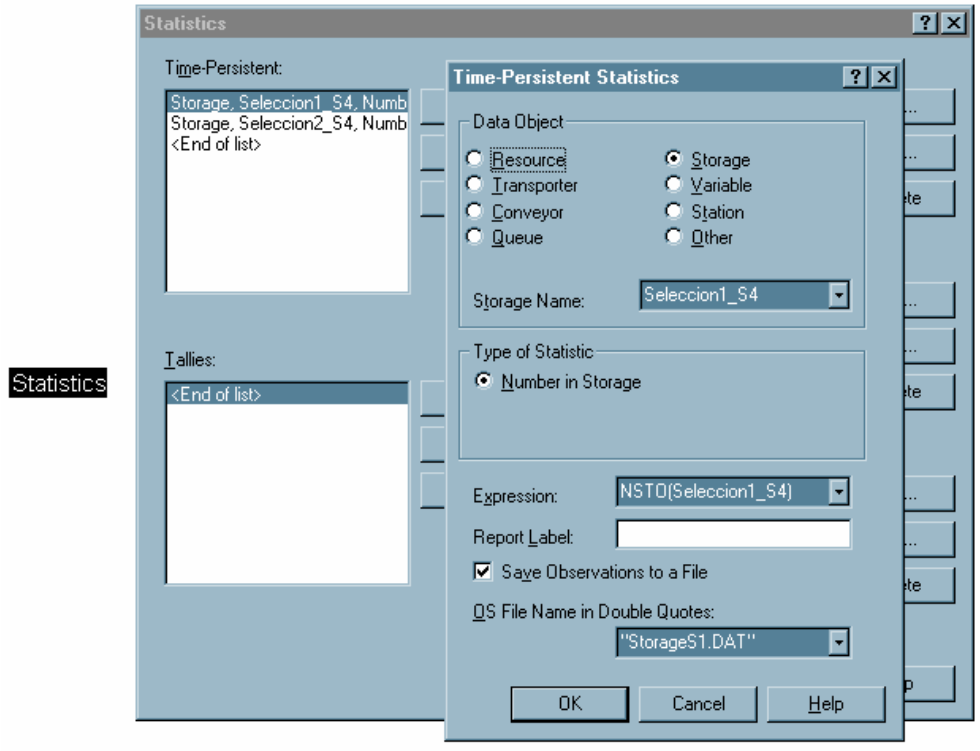


Módulo de Estadísticas.

Este módulo *Statistics* permite guardar algunas variables del sistema en archivos binarios para su posterior estudio estadístico. Así se puede ver a posteriori cómo se han portado determinadas partes del sistema que resultan de interés.

En este caso se va a estudiar el tamaño de las zonas de almacenamiento de salida de los módulos Selección 1 y 2, ya que, mientras se está esperando que lleguen los AGV para recoger piezas, se puede suponer que se irán llenando estas zonas, convirtiéndose en cuellos de botella del sistema.

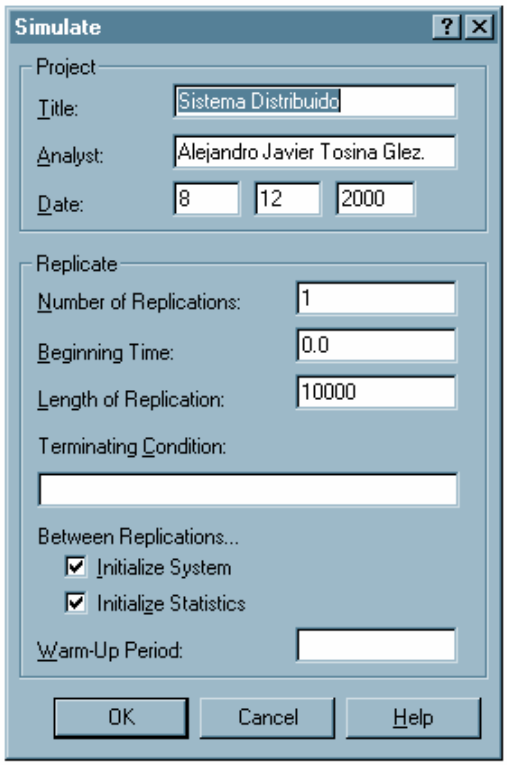
Estos estadísticos se definen de la siguiente forma.



Módulo de Simulación.

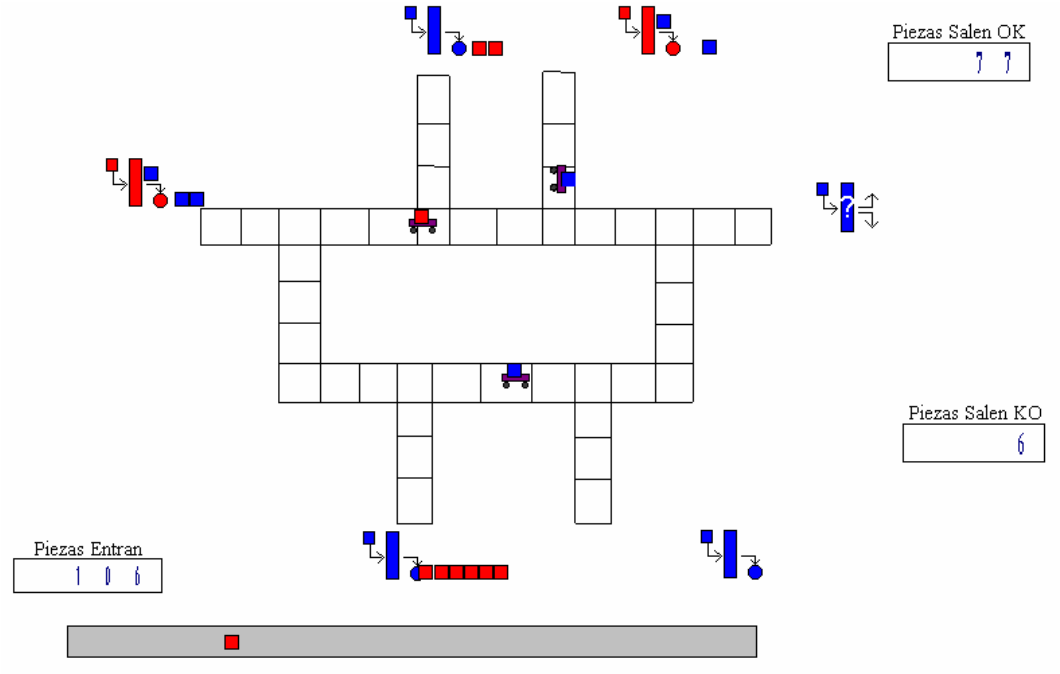
Aquí se definen los parámetros de la simulación, como el tiempo que va a durar, el número de realizaciones de dicha simulación e incluso algún tipo de condición de terminación.

En esta simulación vamos a tener las siguientes condiciones.



Simulación del Sistema.

A continuación se simulará el sistema de producción para comprobar su comportamiento.



Los resultados obtenidos son los siguientes.

ARENA Simulation Results
Alejandro Javier Tosina Glez. - License #8910593

Summary for Replication 1 of 1

Project: Sistema Distribu Run execution date : 11/ 1/2001
Analyst: Alejandro Javier Model revision date: 8/12/2000

Replication ended at time : 10000.0

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
Tiempo en Sistema 1	1260.3	(Insuf)	174.61	2451.7	228
Tiempo en Sistema 2	336.55	(Insuf)	109.23	833.81	158
Proceso1_R_Q Queue Tim	5.9769	2.0449	.00000	70.504	414
Seleccion1_R_Q Queue T	.17764	(Insuf)	.00000	8.4487	293
Inspector_R_Q Queue Ti	78.462	41.167	.00000	475.66	386
Proceso2_R_Q Queue Tim	2.1594	(Insuf)	.00000	57.084	244
Seleccion2_R_Q Queue T	.24712	(Insuf)	.00000	11.156	170
Proceso3_R_Q Queue Tim	2.3069	1.1217	.00000	59.802	409

DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
Proceso3_R Busy	.37644	.03389	.00000	1.0000	1.0000
Proceso2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Proceso2_R Busy	.21070	.03312	.00000	1.0000	.00000
Proceso1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Proceso1_R Busy	.47629	.05226	.00000	1.0000	.00000
# in Seleccion2_R_Q	.00420	(Insuf)	.00000	1.0000	.00000
Inspector_R Busy	.08843	.02570	.00000	2.0000	.00000
AGV Active	3.0000	(Insuf)	3.0000	3.0000	3.0000
AGV Busy	2.9867	(Corr)	.00000	3.0000	3.0000
# in Seleccion1_R_Q	.00520	(Insuf)	.00000	2.0000	.00000
Seleccion2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Seleccion1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
NSTO(SELECCION1_S4)	147.70	(Insuf)	.00000	293.00	293.00
NSTO(SELECCION2_S4)	85.906	(Insuf)	.00000	169.00	169.00
# in Proceso3_R_Q	.09435	(Insuf)	.00000	3.0000	.00000
Seleccion2_R Busy	.06468	.01232	.00000	1.0000	1.0000
Seleccion1_R Busy	.06199	.00929	.00000	1.0000	.00000
# in Proceso2_R_Q	.05269	(Insuf)	.00000	3.0000	.00000
# in Proceso1_R_Q	.24744	(Insuf)	.00000	5.0000	.00000
# Conveying on Cinta T	.34599	.02855	.00000	3.0000	.00000
Inspector_R Available	1.0080	(Insuf)	.00000	2.0000	.00000
# in Inspector_R_Q	3.3690	(Insuf)	.00000	23.000	17.000
Length Conveying on Ci	1.0557	.10405	.00000	10.000	.00000
Proceso3_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000

COUNTERS

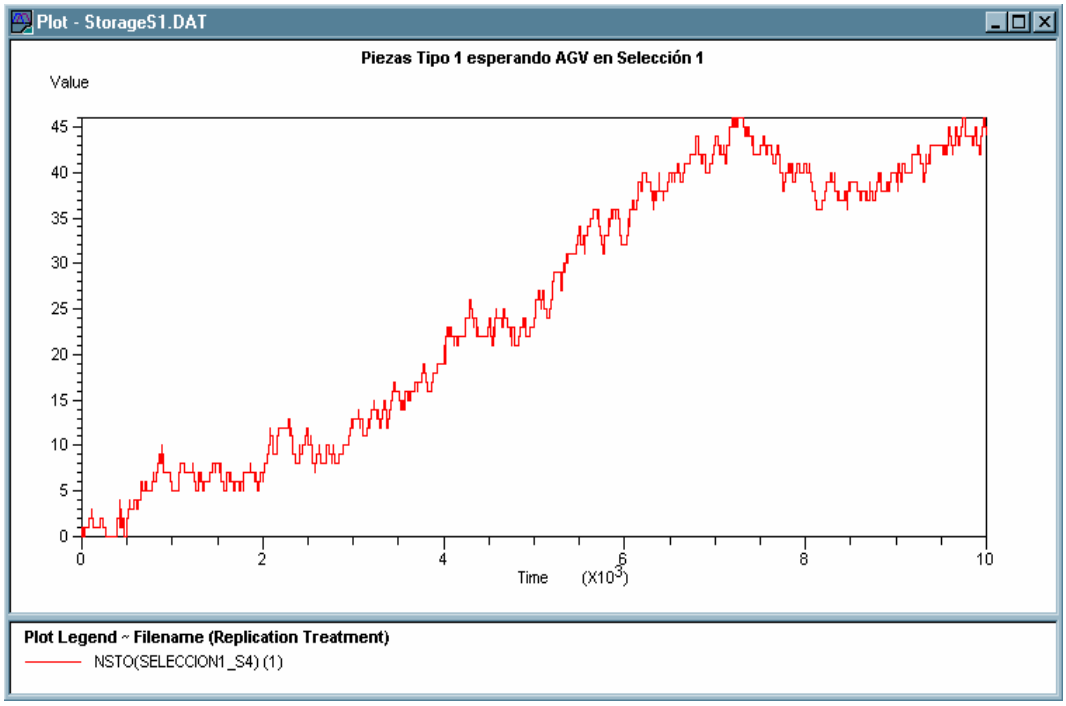
Identifier	Count	Limit
Piezas Salen OK Tipo 1	206	Infinite
Piezas Salen OK Tipo 2	150	Infinite
Piezas Salen KO Tipo 1	22	Infinite
Piezas Salen KO Tipo 2	8	Infinite
Piezas Entran Tipo 1	293	Infinite
Piezas Entran Tipo 2	170	Infinite

Simulation run time: 1.02 minutes.
Simulation run complete.

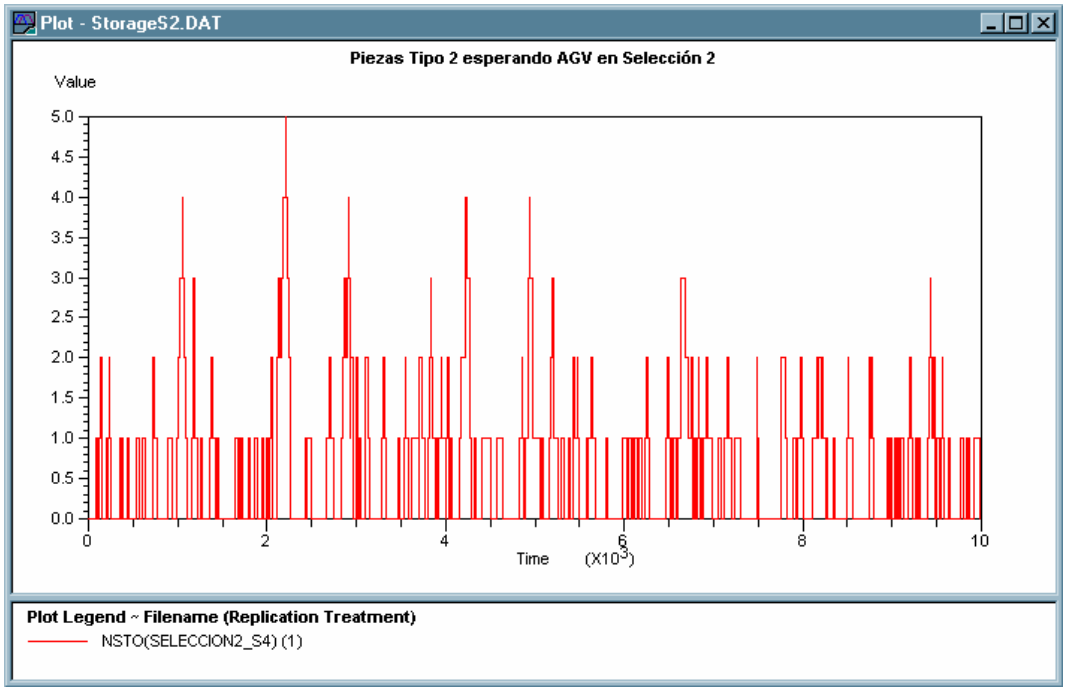
Como se puede comprobar, el sistema está bastante congestionado. Entran en él 293 piezas de tipo 1 y 170 piezas de tipo 2 pero sólo salen 228 piezas de tipo 1 (206 bien y 22 mal) y 158 piezas de tipo 2 (150 bien y 8 mal). Además, el tiempo medio de permanencia en el sistema para las piezas de tipo 1 es de 1260.3 minutos, mientras que las de tipo 2 tienen que esperar 336.55 minutos. Estas cifras, comparadas con los tiempos que manejan los procesos, son demasiado grandes.

Observando los resultados, se puede ver que el principal cuello de botella se encuentra en la red de AGV, puesto que los servidores están bastante desocupados la mayor parte del tiempo (el proceso más ocupado es Proceso1 con una carga del 47.629%).

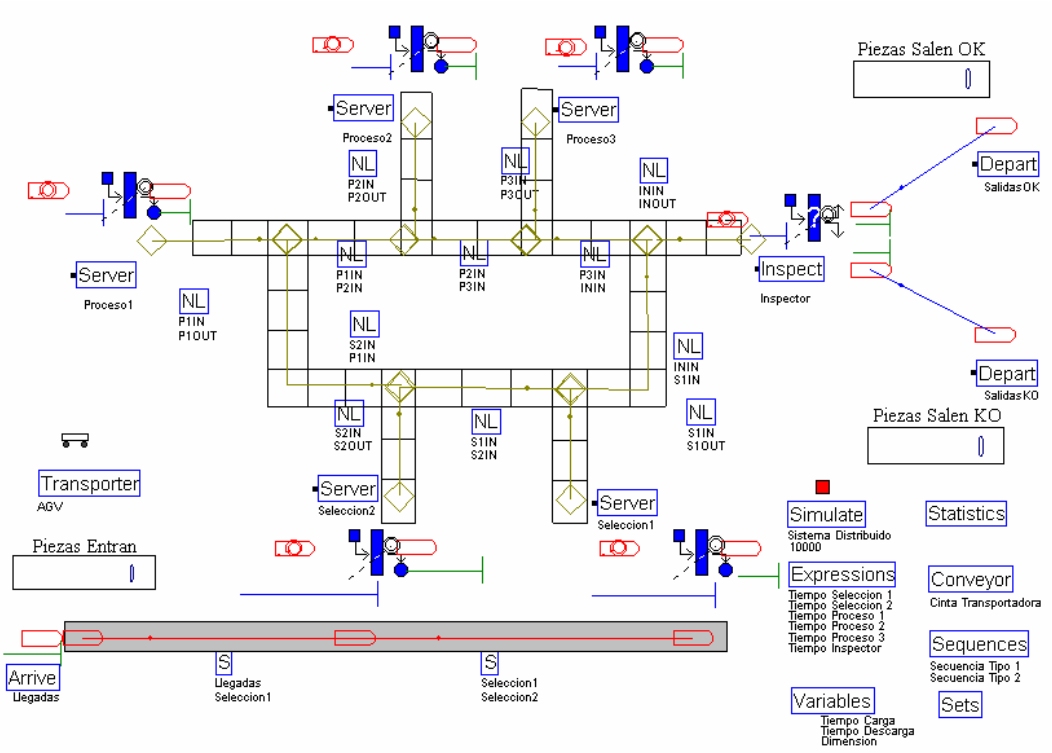
El principal problema consiste en que normalmente hay muchas piezas en los *Storages* de salida de las estaciones esperando que se les asigne un AGV. Así se puede comprobar que los *Storages* de salida de los módulos de selección están muy saturados y, conforme pasa el tiempo, se saturan más, con lo cual llegará a colapsarse el sistema.



Otro problema que se ve en la simulación consiste en que, al estar el módulo Selección 1 detrás de Selección 2 en el orden en que llegan los vehículos de transporte, las piezas de tipo 1, que son las más numerosas, se ven notablemente perjudicadas. Esto es debido a que cuando un AGV se libera de transportar piezas entre las estaciones de proceso, se dirige a coger piezas de selección. Así pués, al estar antes la estación Selección 2, con una sola pieza que tenga esperando, aunque la estación Selección 1 tenga muchas, el AGV se dirigirá a la 2. De esta forma se observa que el *Storage* de salida de la estación Selección 2 está muchísimo más descargado que el de la estación Selección 1.



Una solución que se puede aportar, consiste en cambiar el orden de las estaciones en la red, anteponiendo al paso de los vehículos automáticos aquellas estaciones cuya afluencia de entidades es más numerosa. De esta forma se colocarán las estaciones en el siguiente orden: Selección1, Selección2, Proceso1, Proceso2 y Proceso3, siguiendo la dirección de las agujas del reloj.



Los resultados de la simulación de este sistema modificado son los siguientes.

```
                    ARENA Simulation Results
            Alejandro Javier Tosina Glez. - License #8910593

                  Summary for Replication 1 of 1

Project:  Sistema Distribu                    Run execution date :  16/ 1/2001
Analyst:  Alejandro Javier                    Model revision date:   8/12/2000

Replication ended at time      : 10000.0

                              TALLY VARIABLES

Identifier              Average    Half Width  Minimum    Maximum    Observations
-------------------------------------------------------------------------------

Tiempo en Sistema 1     316.81     (Insuf)     80.042     801.68         271
Tiempo en Sistema 2     442.10     (Insuf)     124.81     1211.4         184
Seleccion1_R_Q Queue T  .08347     (Insuf)     .00000     5.2663         303
Proceso1_R_Q Queue Tim  7.5191     2.1346      .00000     66.023         488
Inspector_R_Q Queue Ti  78.176     45.812      .00000     478.51         455
Proceso2_R_Q Queue Tim  2.0780     (Insuf)     .00000     50.833         295
Seleccion2_R_Q Queue T  .43377     (Insuf)     .00000     15.926         199
Proceso3_R_Q Queue Tim  2.2559     .98251      .00000     40.378         481
```

DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
Proceso3_R Busy	.41119	.03096	.00000	1.0000	1.0000
Proceso2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Proceso2_R Busy	.24803	.04062	.00000	1.0000	1.0000
Proceso1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Proceso1_R Busy	.54786	.06177	.00000	1.0000	1.0000
# in Seleccion2_R_Q	.00863	(Insuf)	.00000	1.0000	.00000
Inspector_R Busy	.11834	.03437	.00000	2.0000	.00000
AGV Active	3.0000	(Insuf)	3.0000	3.0000	3.0000
AGV Busy	2.9792	(Corr)	.00000	3.0000	3.0000
# in Seleccion1_R_Q	.00253	(Insuf)	.00000	2.0000	.00000
Seleccion2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Seleccion1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
NSTO(SELECCION1_S4)	1.1598	.26946	.00000	6.0000	4.0000
NSTO(SELECCION2_S4)	3.6224	(Corr)	.00000	15.000	8.0000
# in Proceso3_R_Q	.10851	(Corr)	.00000	3.0000	.00000
Seleccion2_R Busy	.08223	.01876	.00000	1.0000	.00000
Seleccion1_R Busy	.05662	.00991	.00000	1.0000	.00000
# in Proceso2_R_Q	.06130	(Insuf)	.00000	3.0000	.00000
# in Proceso1_R_Q	.36982	.10447	.00000	4.0000	2.0000
# Conveying on Cinta T	.38472	.04516	.00000	4.0000	.00000
Inspector_R Available	1.0080	(Insuf)	.00000	2.0000	.00000
# in Inspector_R_Q	4.0235	1.8679	.00000	26.000	23.000
Length Conveying on Ci	1.1982	.15217	.00000	14.000	.00000
Proceso3_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000

COUNTERS

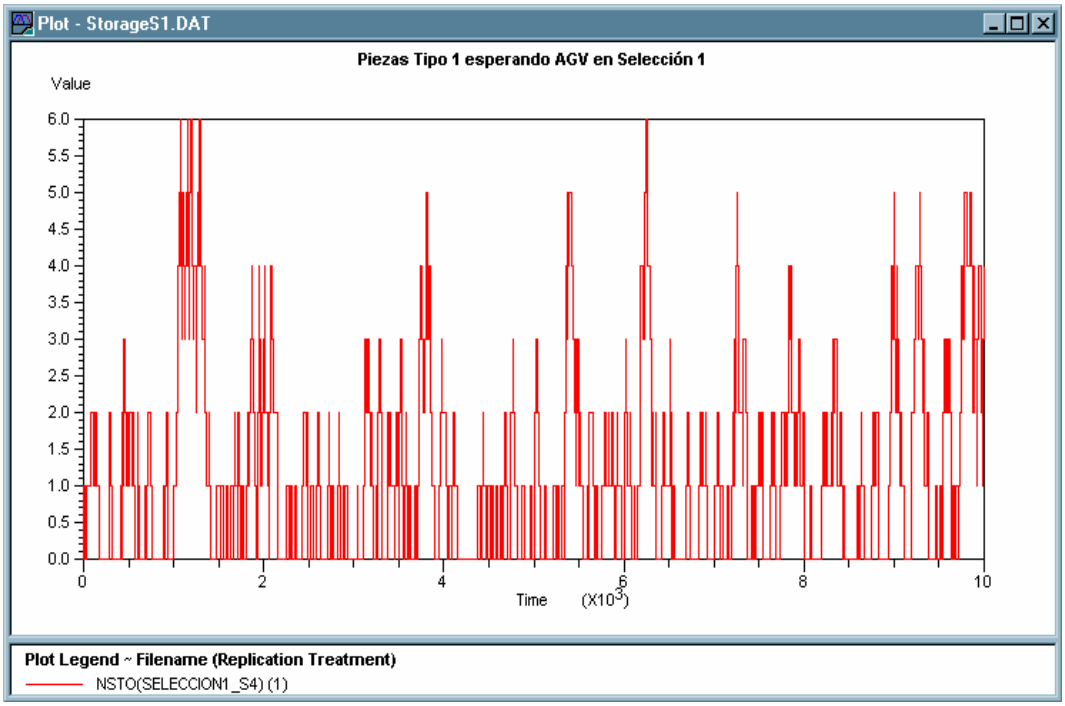
Identifier	Count	Limit
Piezas Salen OK Tipo 1	245	Infinite
Piezas Salen OK Tipo 2	166	Infinite
Piezas Salen KO Tipo 1	26	Infinite
Piezas Salen KO Tipo 2	18	Infinite
Piezas Entran Tipo 1	303	Infinite
Piezas Entran Tipo 2	199	Infinite

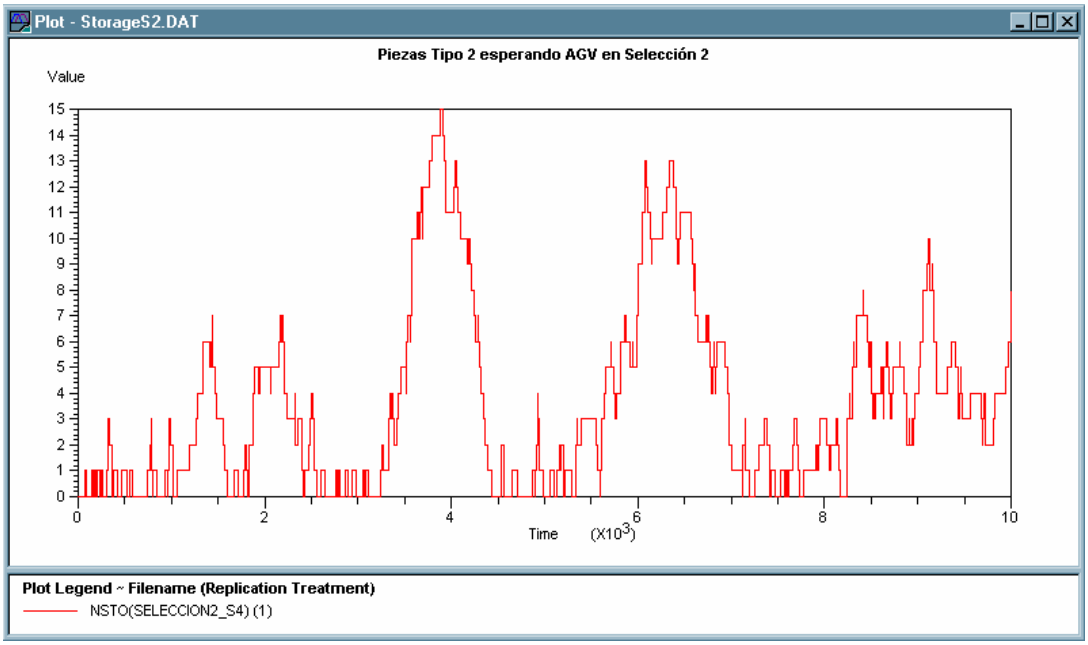
Simulation run time: 0.85 minutes.
Simulation run complete.

Se puede apreciar que los resultados de la modificación realizada son bastante buenos. El tiempo que tardan las piezas de tipo 1 en salir del sistema ha disminuido en cuatro veces. Mientras que antes tardaban 1260.3 minutos en salir, ahora tardan 316.81 minutos. Sin embargo las piezas de tipo 2 duraban en el sistema antiguo 336.55 minutos, mientras que ahora duran 442.10 minutos. Se observa que el empeoramiento que se produce en los tiempos de las piezas de tipo 2 es notablemente inferior a la mejora producida en los tiempos de las piezas de tipo 1 que, al ser más numerosas, producirán una mejora sustancial en el rendimiento global del sistema.

La mejora producida se puede ver mejor con el hecho de que, mientras antes salían 206 piezas de tipo 1 y 150 piezas de tipo 2, ahora salen del sistema 245 piezas de tipo 1 y 166 piezas de tipo 2, con lo cual la producción aumenta de forma considerable, incluyendo las piezas de tipo 2, que han sido perjudicadas por la nueva distribución del espacio en la planta.

De forma gráfica se puede comprobar el cambio producido en los *Storages* de salida de las estaciones Selección, lo que nos dará una idea de la mejora conseguida, no solo en cuanto a número de entidades esperando transporte sino en cuanto a estabilidad del sistema, puesto que no sucede como en el caso anterior, en el que el número de entidades en el Storage 1 se disparaba llegando a saturar el sistema, sino que el número de piezas esta vez es muy limitado y no muestra ninguna tendencia de crecimiento, salvo variaciones puntuales en torno a un nivel de equilibrio.





Otro aspecto a tener en cuenta resulta ser el factor de carga de los procesos. En este nuevo sistema están más ocupados que en el antiguo, lo que repercute en una mayor utilización y, por lo tanto, en un mayor rendimiento de los equipos.

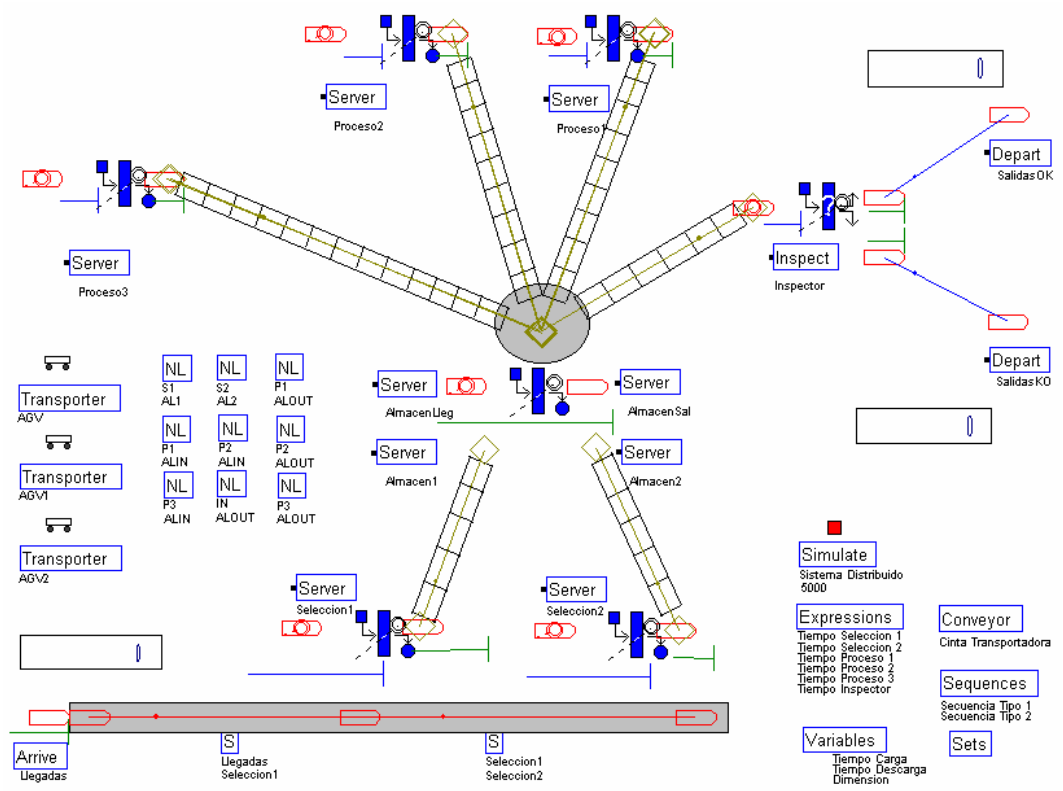
Proceso	Carga sistema antiguo	Carga sistema nuevo
Proceso1	0.47629	0.54786
Proceso2	0.21079	0.24803
Proceso3	0.37644	0.41119

De esta forma se ha comprobado la importancia que en los sistemas de fabricación flexible tiene la correcta distribución de los elementos a fin de optimizar el rendimiento del sistema y, como consecuencia de esto, aumentar el beneficio.

Sistema Almacén Central.

Modelo del Sistema.

El modelo de este sistema que se va a estudiar a continuación es el siguiente.

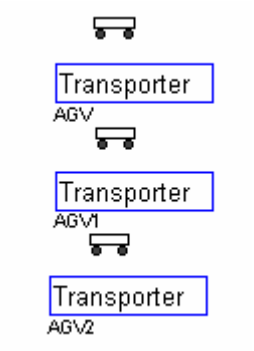


Se puede apreciar que, una vez seleccionadas las piezas, todos los movimientos se producen dentro de una red de transporte en estrella, cuyo nodo central es el sistema de almacenamiento centralizado que sustituye a la red de distribución del sistema anterior.

En este caso se tienen tres vehículos de guiado automático (AGV) que irán transportando las piezas de ambos tipos a través de una red de enlaces bidireccionales:

- El vehículo AGV1 se encargará de transportar las piezas de tipo 1 desde la estación Selección 1 hasta el Almacén Central.
- El vehículo AGV2 transportará las piezas de tipo 2 desde Selección 2 hasta el Almacén Central.
- El vehículo AGV es el encargado de mover las piezas entre el Almacén Central y los distintos procesos, así como de llevar las piezas hacia el módulo de Inspección.

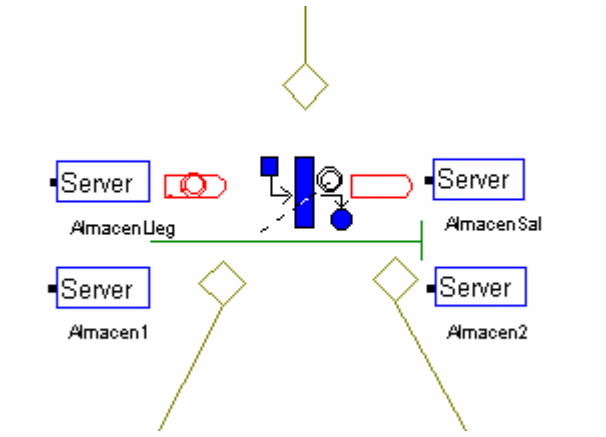
Hay que tener en cuenta que se utilizan tres *Transporters* en vez de uno sólo con tres unidades. Esto se ha hecho así para permitir que los AGV encargados de transportar entidades entre los módulos de Selección y el Almacén Central, una vez que han transportado su mercancía, puedan volver a estos módulos iniciales de Selección, de los que partieron, y así ahorrar un tiempo valioso a la hora de recoger una nueva entidad, ya que no tendrán que esperar a que se les reclame, pues ya estarán allí. Con un solo *Transporter* habría que definir una posición de retorno común para los tres, lo cual sería inviable debido a problemas de colisiones y de conectividad de rutas.



Otro detalle a tener en cuenta consiste en el diseño del Almacén Central, que está formado por cuatro módulos, tres de entrada y uno de salida. Cada módulo es un *Server* de tiempo de proceso nulo cuya funcionalidad queda descrita de la siguiente forma:

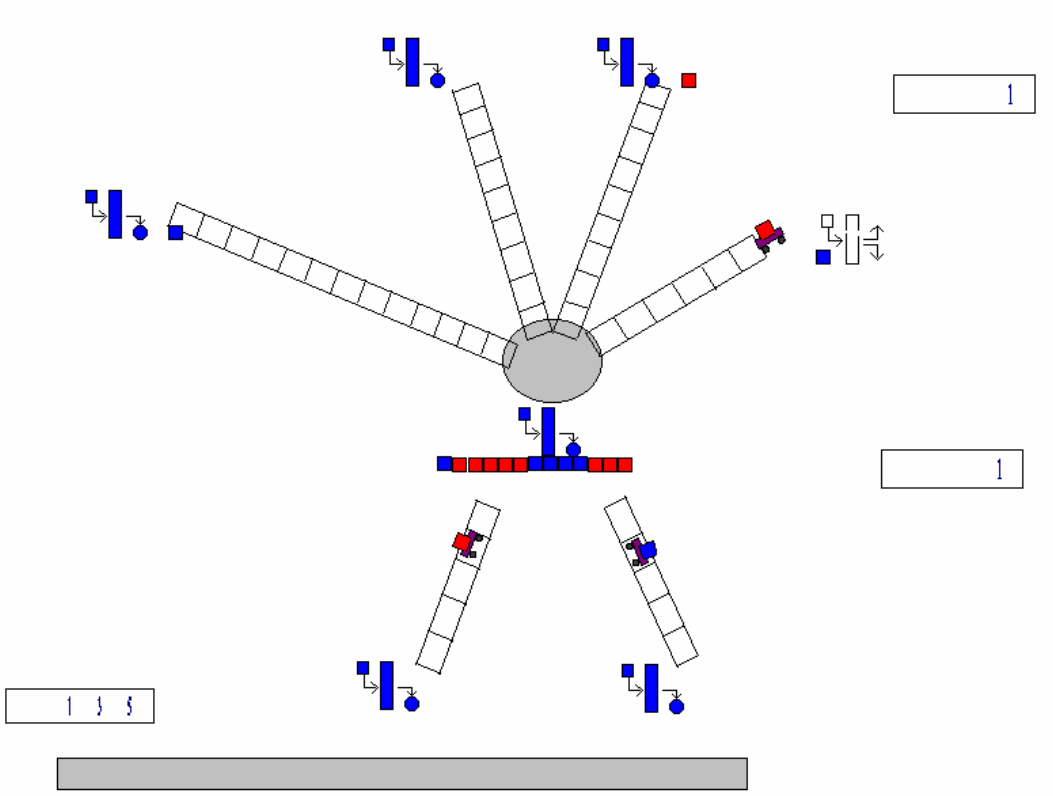
- *Almacen1*: Este módulo sirve para recibir las piezas de tipo 1 que van llegando procedentes del *Server* Selección 1. Descarga las piezas y las envía directamente a través de un Route con retardo cero hasta el módulo AlmacénSal.
- *Almacen2*: Este módulo es idéntico a Almacén 1 pero realiza su función con las piezas de tipo 2.
- *AlmacenLleg*: Por este módulo van llegando las entidades procedentes de algún módulo de Proceso. Realiza las mismas funciones que los módulos anteriores.
- *AlmacenSal*: Este módulo puede decirse que es el Almacén Central real. Almacena las piezas en su *Storage* correspondiente y se encarga de distribuirlas hacia los módulos Proceso1, 2, 3 y hacia el módulo de Inspección.

Esto se ha realizado así porque era necesario asociar tres nodos de tres redes distintas al Almacén Central y Arena 3.0 no permite asociarlos a la misma estación. De esta forma se consigue engañar al sistema asociando un nodo a cada estación pero, al tener estos *Server* tiempo de ejecución nulo y estar conectados directamente con retardo cero, en realidad simula una sola estación conectada a tres redes simultáneamente.



Simulación del sistema.

Se simula el sistema y se obtienen los siguientes resultados.



ARENA Simulation Results
Alejandro Javier Tosina Glez. - License #8910593

Summary for Replication 1 of 1

Project: Sistema Distribu
Analyst: Alejandro Javier

Run execution date : 18/ 1/2001
Model revision date: 8/12/2000

Replication ended at time : 10000.0

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
AlmacenLleg_R_Q Queue	.00000	.00000	.00000	.00000	427
Tiempo en Sistema 1	4711.3	(Insuf)	653.87	8742.2	25
Tiempo en Sistema 2	3994.3	(Insuf)	329.98	8026.2	22
Proceso1_R_Q Queue Tim	.18312	(Insuf)	.00000	14.393	208
Seleccion1_R_Q Queue T	.09044	(Insuf)	.00000	4.3300	276
Almacen1_R_Q Queue Tim	.00000	(Insuf)	.00000	.00000	276
Inspector_R_Q Queue Ti	67.512	(Insuf)	.00000	458.90	47
Proceso2_R_Q Queue Tim	.00000	(Insuf)	.00000	.00000	75
Seleccion2_R_Q Queue T	.19330	(Insuf)	.00000	13.234	212
AlmacenSal_R_Q Queue T	.00000	.00000	.00000	.00000	914
Almacen2_R_Q Queue Tim	.00000	(Insuf)	.00000	.00000	211
Proceso3_R_Q Queue Tim	.37411	(Insuf)	.00000	25.997	144

DISCRETE-CHANGE VARIABLES

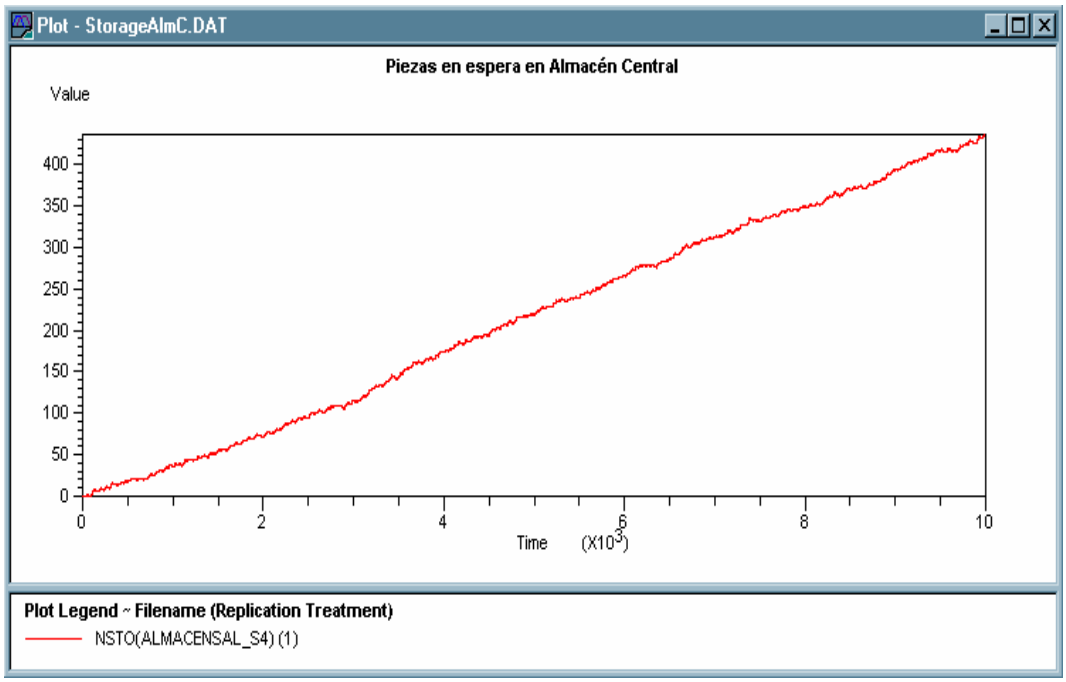
Identifier	Average	Half Width	Minimum	Maximum	Final Value
Proceso3_R Busy	.13232	(Insuf)	.00000	1.0000	.00000
Proceso2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Proceso2_R Busy	.03085	(Insuf)	.00000	1.0000	.00000
Proceso1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Proceso1_R Busy	.10637	.01717	.00000	1.0000	.00000
AGV2 Busy	.21826	.03140	.00000	1.0000	.00000
AGV1 Busy	.28534	.03862	.00000	1.0000	.00000
AlmacenSal_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
AGV Busy	.99747	(Corr)	.00000	1.0000	1.0000
AGV Active	1.0000	(Insuf)	1.0000	1.0000	1.0000
# in Seleccion2_R_Q	.00410	(Insuf)	.00000	2.0000	.00000
Inspector_R Busy	.01296	(Insuf)	.00000	2.0000	.00000
# in Seleccion1_R_Q	.00250	(Insuf)	.00000	1.0000	.00000
# in Almacen2_R_Q	.00000	(Insuf)	.00000	.00000	.00000
Seleccion2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
AlmacenLleg_R Busy	.00000	.00000	.00000	1.0000	.00000
# in Almacen1_R_Q	.00000	(Insuf)	.00000	.00000	.00000
Seleccion1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
AlmacenSal_R Busy	.00000	.00000	.00000	1.0000	.00000
NSTO(SELECCION1_S4)	.06716	.02656	.00000	2.0000	.00000
Almacen2_R Busy	.00000	(Insuf)	.00000	1.0000	.00000
Almacen2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
NSTO(SELECCION2_S4)	.03109	.01131	.00000	2.0000	.00000
NSTO(ALMACENSAL_S4)	215.73	(Corr)	.00000	436.00	435.00
Almacen1_R Busy	.00000	(Insuf)	.00000	1.0000	.00000
Almacen1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
# in Proceso3_R_Q	.00539	(Insuf)	.00000	1.0000	.00000
Seleccion2_R Busy	.09056	.01659	.00000	1.0000	1.0000
Seleccion1_R Busy	.05438	.00879	.00000	1.0000	.00000
# in Proceso2_R_Q	.00000	(Insuf)	.00000	.00000	.00000
# in Proceso1_R_Q	.00381	(Insuf)	.00000	2.0000	.00000
# Conveying on Cinta T	.38161	.03661	.00000	4.0000	.00000
AlmacenLleg_R Availabl	1.0000	(Insuf)	1.0000	1.0000	1.0000
AGV2 Active	1.0000	(Insuf)	1.0000	1.0000	1.0000
Inspector_R Available	1.0080	(Insuf)	.00000	2.0000	.00000
AGV1 Active	1.0000	(Insuf)	1.0000	1.0000	1.0000
# in Inspector_R_Q	.41313	(Insuf)	.00000	4.0000	4.0000
Length Conveying on Ci	1.2173	.13253	.00000	12.000	.00000
# in AlmacenLleg_R_Q	.00000	(Insuf)	.00000	.00000	.00000
# in AlmacenSal_R_Q	.00000	(Insuf)	.00000	.00000	.00000
Proceso3_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000

COUNTERS

Identifier	Count	Limit
Piezas Salen OK Tipo 1	25	Infinite
Piezas Salen OK Tipo 2	22	Infinite
Piezas Salen KO Tipo 1	0	Infinite
Piezas Salen KO Tipo 2	0	Infinite
Piezas Entran Tipo 1	276	Infinite
Piezas Entran Tipo 2	212	Infinite

Simulation run time: 1.77 minutes.
Simulation run complete.

Como se puede comprobar, los resultados no son satisfactorios. De 276 piezas de tipo 1 que entran en el sistema, salen bien 25 y mal 0. De 212 piezas de tipo2 que entran en el sistema, salen bien 22, mientras que mal no sale ninguna. El tiempo medio de permanencia en el sistema de las piezas de tipo 1 es de 4711.3 minutos y el de las piezas de tipo 2 de 3994.3 minutos. Además, el rendimiento de los equipos es muy bajo, pues toda la carga la soporta el Almacén Central.



Elemento	Ocupación
AGV	99.747%
AGV1	28.534%
AGV2	21.826%
Proceso 1	10.637%
Proceso 2	03.085%
Proceso 3	13.232%

Este sistema, sin embargo, puede ser mejorado observando algunos aspectos de la simulación.

Primero se puede apreciar que el AGV que se encarga de transportar las piezas entre el Almacén Central y los Procesos está bastante cargado, mientras que los otros dos AGV tienen un grado de ocupación bastante reducido.

Otro aspecto importante a tener en cuenta es el de las colas de espera para conseguir un vehículo. En Selección 1 y Selección 2 son muy pequeñas y, a veces, inexistentes, mientras que en al Almacén Central tiende a saturarse, con lo cual llegará un momento en el que el sistema se colapsará.

Se va a modificar el sistema para descargar de alguna forma al AGV de las máquinas de proceso y, así, intentar acelerar la salida de piezas del Almacén. De esta forma se evitará su saturación. Esto se va a realizar dedicando dos AGV a los trayectos entre dicho Almacén y los Procesos, uno para Proceso 1 e Inspector (AGV1) y otro para Proceso 2 y Proceso 3 (AGV2), mientras que dejamos a un AGV solo el trabajo de recoger las piezas de los módulos de Selección.

Los resultados obtenidos son los siguientes.

ARENA Simulation Results
Alejandro Javier Tosina Glez. - License #8910593

Summary for Replication 1 of 1

Project: Sistema Distribu Run execution date : 19/ 1/2001
Analyst: Alejandro Javier Model revision date: 8/12/2000

Replication ended at time : 10000.0

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
AlmacenLleg_R_Q Queue	.00000	.00000	.00000	.00000	506
Tiempo en Sistema 1	3007.5	(Insuf)	115.60	6184.2	96
Tiempo en Sistema 2	2689.0	(Insuf)	144.39	5059.5	97
Seleccion1_R_Q Queue T	.08669	(Insuf)	.00000	6.6296	310
Proceso1_R_Q Queue Tim	.20513	(Corr)	.00000	17.981	370
AlmacenSal1_R_Q Queue	.00000	.00000	.00000	.00000	723
Almacen1_R_Q Queue Tim	.00000	.00000	.00000	.00000	369
Inspector_R_Q Queue Ti	81.774	(Insuf)	.00000	478.16	193
Seleccion2_R_Q Queue T	.28429	(Insuf)	.00000	10.421	198
Proceso2_R_Q Queue Tim	.07945	(Insuf)	.00000	6.1643	175
AlmacenSal2_R_Q Queue	.00000	.00000	.00000	.00000	613
Almacen2_R_Q Queue Tim	.00000	.00000	.00000	.00000	461
Proceso3_R_Q Queue Tim	3.2202	(Insuf)	.00000	58.161	288

DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
AlmacenSal2_R Busy	.00000	.00000	.00000	1.0000	.00000
Proceso2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Proceso3_R Busy	.29562	.03928	.00000	1.0000	.00000
# in AlmacenSal2_R_Q	.00000	(Insuf)	.00000	.00000	.00000
AlmacenSal1_R Busy	.00000	.00000	.00000	1.0000	.00000
Proceso1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Proceso2_R Busy	.07214	.01023	.00000	1.0000	.00000
# in AlmacenSal1_R_Q	.00000	(Insuf)	.00000	.00000	.00000
Proceso1_R Busy	.19425	.01826	.00000	1.0000	.00000
NSTO(ALMACENSAL1_S4)	68.854	(Corr)	.00000	154.00	154.00
AGV2 Busy	.99281	(Corr)	.00000	1.0000	1.0000
NSTO(ALMACENSAL2_S4)	80.923	(Corr)	.00000	152.00	150.00
AGV1 Busy	.98928	(Corr)	.00000	1.0000	1.0000
Inspector_R Busy	.05123	.01580	.00000	2.0000	.00000
# in Seleccion2_R_Q	.00563	(Insuf)	.00000	2.0000	.00000
AGV Active	1.0000	(Insuf)	1.0000	1.0000	1.0000
AGV Busy	.55647	.05020	.00000	1.0000	1.0000
AlmacenSal2_R Availabl	1.0000	(Insuf)	1.0000	1.0000	1.0000
# in Seleccion1_R_Q	.00269	(Insuf)	.00000	1.0000	.00000
AlmacenSal1_R Availabl	1.0000	(Insuf)	1.0000	1.0000	1.0000
AlmacenLleg_R Busy	.00000	.00000	.00000	1.0000	.00000
Seleccion2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
# in Almacen2_R_Q	.00000	(Insuf)	.00000	.00000	.00000
Seleccion1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
# in Almacen1_R_Q	.00000	(Insuf)	.00000	.00000	.00000
Almacen2_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Almacen2_R Busy	.00000	.00000	.00000	1.0000	.00000
Almacen1_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000
Almacen1_R Busy	.00000	.00000	.00000	1.0000	.00000
Seleccion2_R Busy	.07160	.01294	.00000	1.0000	.00000
# in Proceso3_R_Q	.09274	(Insuf)	.00000	4.0000	.00000
# in Proceso2_R_Q	.00139	(Insuf)	.00000	1.0000	.00000
Seleccion1_R Busy	.06271	.01095	.00000	1.0000	.00000
AlmacenLleg_R Availabl	1.0000	(Insuf)	1.0000	1.0000	1.0000
# Conveying on Cinta T	.38581	.03929	.00000	4.0000	.00000
# in Proceso1_R_Q	.00759	(Insuf)	.00000	1.0000	.00000
Inspector_R Available	1.0080	(Insuf)	.00000	2.0000	.00000
AGV2 Active	1.0000	(Insuf)	1.0000	1.0000	1.0000
# in AlmacenLleg_R_Q	.00000	(Insuf)	.00000	.00000	.00000
Length Conveying on Ci	1.1958	.13724	.00000	14.000	.00000
# in Inspector_R_Q	1.7027	(Insuf)	.00000	16.000	6.0000
AGV1 Active	1.0000	(Insuf)	1.0000	1.0000	1.0000
Proceso3_R Available	1.0000	(Insuf)	1.0000	1.0000	1.0000

COUNTERS

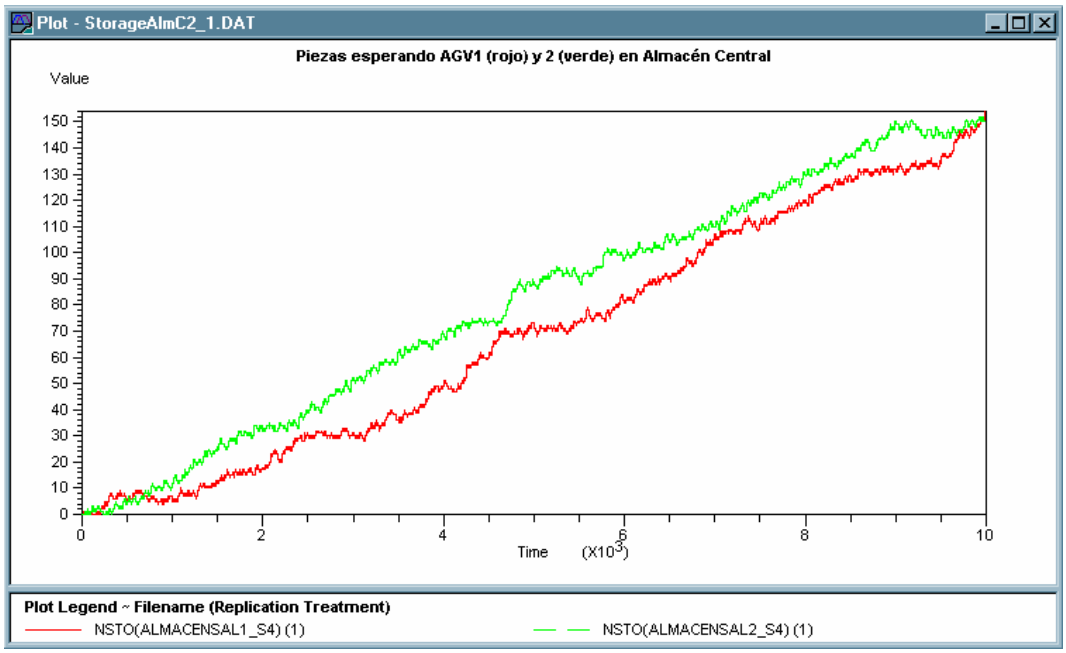
Identifier	Count	Limit
Piezas Salen OK Tipo 1	84	Infinite
Piezas Salen OK Tipo 2	87	Infinite
Piezas Salen KO Tipo 1	12	Infinite
Piezas Entran Tipo 1	310	Infinite
Piezas Salen KO Tipo 2	10	Infinite
Piezas Entran Tipo 2	198	Infinite

Simulation run time: 0.48 minutes.
Simulation run complete.

Se puede comprobar que el comportamiento del sistema experimenta una leve mejoría, pero insuficiente para competir con el Sistema Distribuido.

Ahora salen 84 piezas buenas de tipo 1, cuando llegaron 310, y 87 piezas buenas de tipo 2, cuando las que entraron fueron 198. El tiempo medio de permanencia en el sistema es de 3007.5 minutos para las piezas de tipo 1 y de 2689.0 minutos para las de tipo 2. Tampoco mejoran de manera significativa los rendimientos de los distintos elementos del sistema.

El final de este sistema será el colapso, debido a no poder resistir el ritmo de saturación que va alcanzando el Almacén Central.



Comparativa de ambos sistemas.

Una vez estudiados ambos sistemas por separado cabe sacar algunas conclusiones.

Evidentemente el Sistema Distribuido es mucho mejor que el basado en un Almacén Central en todos los aspectos, puesto que el número de piezas que obtiene como resultado en el tiempo de simulación es ampliamente superior, el tiempo de permanencia en el sistema es muy reducido y el rendimiento de los elementos utilizados en el proceso de fabricación es mayor, con lo cual el sistema es más eficiente y aprovecha mejor los recursos empleados.

El principal motivo de esta superioridad consiste en que, al ser una red distribuida en la cual todos los vehículos realizan todas las tareas, el trabajo se reparte mejor que en el sistema centralizado. Esto se observa bien en el análisis de resultados, puesto que en el Sistema de Almacén Central había vehículos que tenían una ocupación muy baja, mientras que otros la tenían muy alta. Esto sucedía porque los AGV están dedicados a partes concretas de la red, así que, aunque uno de ellos esté desbordado y los demás libres, éstos no acudirán a ayudarle.

En el sistema distribuido, por el contrario, todos los AGV realizan el mismo trabajo, con lo cual la carga de la red está más repartida y no hay un cuello de botella tan crítico. Sin embargo, es importante destacar que en este tipo de sistemas el ordenamiento del tráfico de la red de transporte es fundamental, debiéndose potenciar aquéllas rutas más transitadas, en detrimento de las que lo son menos.

Sin duda, en el caso centralizado, si se dispusiera de un AGV para cada enlace entre estaciones, el sistema sería mejor que el distribuido en cuanto a nivel de producción, puesto que cada vehículo se encargaría de su enlace y ninguno estaría saturado por el ritmo de llegada de piezas. No obstante, los recursos estarían infrautilizados, más de lo que en el caso presente se muestra.

Como conclusión a este análisis, cabe resaltar que para el sistema que ha sido sometido a estudio es claramente mejor una topología de red de transporte distribuida, la cual ha demostrado ser más eficaz.

Conclusiones.

Para concluir, es necesario resaltar algunos aspectos importantes que han sido abordados a lo largo del presente proyecto.

En primer lugar se ha visto la gran utilidad que tiene la Simulación a la hora de estudiar sistemas reales. No solo permite un análisis de los mismos con más o menos detalle sino que, además, facilita su optimización e incluso su creación en condiciones óptimas cuando todavía no existen. Todo esto a un coste cada vez más reducido.

Se han estudiado los Lenguajes de Simulación, los cuales constituyen una herramienta necesaria, sin la cual las técnicas de Simulación modernas no pueden implementarse. Además, se han resaltado las enormes ventajas, sobre todo en cuanto a facilidad de uso se refiere, que presentan las herramientas visuales (VIMS) sobre otros lenguajes y se ha profundizado en el estudio de una de las más conocidas, Arena 3.0.

La parte final del proyecto se ha centrado en el estudio de los Sistemas de Fabricación Flexible (FMS) y se han obtenido unas conclusiones interesantes como la mayor idoneidad del reparto del trabajo y la ventaja de la optimización de las tareas más numerosas a fin de mejorar un sistema en su conjunto.

Por último cabe resaltar la importancia y el interés de este proyecto al estudiar una herramienta, la Simulación, que permite analizar y mejorar sistemas reales de todo tipo, siendo de gran utilidad en todos los campos, desde la Industria y las Telecomunicaciones hasta la Economía, la Política y la Sociedad.

Bibliografía.

BANKS J., CARSON J.S.; “Discrete-Event System Simulation”; Prentice Hall, 1984.

CARRIE A.; “Simulation of Manufacturing Systems”; Wiley, 1988.

HOOVER S.V., PERRY R.F.; “Simulation. A Problem-Solving Approach”; Addison Wesley, 1989.

KELTON W.D., SADOWSKI R.P., SADOWSKI D.A.; “Simulation with Arena”; MacGraw-Hill, 1998.

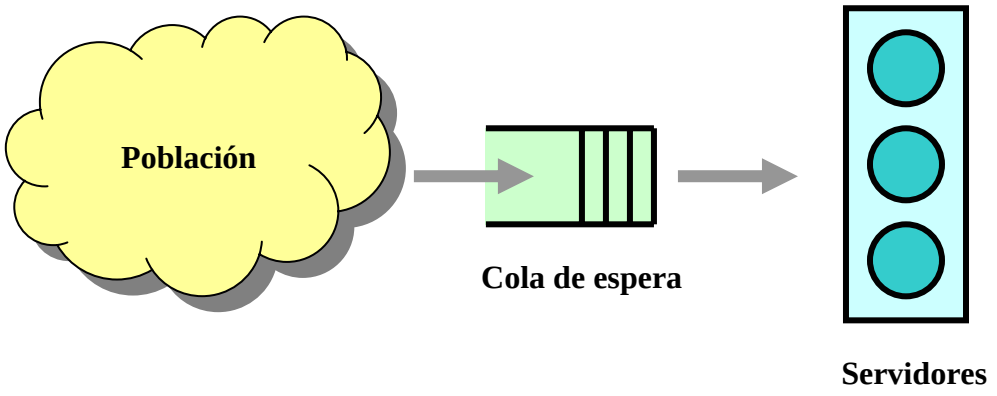
MALEKI R.A.; “Flexible Manufacturing Systems. The Technology and Management”; Prentice Hall 1991.

PIDD M.; “Computer Simulation in Management Science”; Wiley, 1998.

Anexo I: Conceptos Generales de Teoría de Colas.

La Teoría de Colas es una herramienta matemática que nos permite analizar sistemas de colas. Estos tipos de sistemas tienen tres componentes:

- Población: Usuarios que quieren utilizar el sistema.
- Cola: Lugar donde esperan las peticiones de la población mientras son atendidas.
- Servidores: Ofrecen el servicio que demandan los usuarios de la población.



Los sistemas de colas pueden ser de dos tipos:

- Abiertos: Los usuarios entran en el sistema, esperan en la cola, son atendidos y salen.
- Cerrados: Los usuarios pertenecen al sistema. Después de ser servidos vuelven a incorporarse a la población.

Hay una serie de parámetros que identifican un sistema de colas y permiten clasificarlos para proporcionarles su tratamiento matemático adecuado.

- Proceso de Llegada: Representa el comportamiento de los usuarios a la hora de solicitar el servicio.

G	General. Sin limitación.
M	Exponencial. Proceso de Poissón.
D	Determinista.
E_k	Erlang-k.

- Tamaño de la cola: Puede ser finita, infinita o no existir.
- Número de servidores.
- Tiempo de servicio (en distribución).

G	General. Sin limitación.
M	Exponencial. Proceso de Poissón.
D	Determinista.
E_k	Erlang-k.

- Tamaño de la fuente.
- Disciplina de la cola: Indica el orden de los usuarios para darles servicio.

FCFS/FIFO	Primero en llegar es el primero en servirse.
LCFS	Primero va el último que llega. Puede ser con desalojo (con reinicio o continuación) o sin desalojo.
Round-Robin RR	Desalojo con continuación. Se dan cuantos de tiempo de proceso de forma periódica.

Para resumir todos estos parámetros se utiliza la notación Kendall, que está compuesta por una serie de letras que identifican el sistema de colas en cuestión.

A/B/C/D/E/F

A	Proceso de llegada.
B	Tiempo de servicio.
C	Número de servidores.
D	Tamaño del sistema (cola + servidores).
E	Tamaño de la población.
F	Disciplina de cola.

Con todas estas ideas y notaciones, lo que se pretende es obtener una serie de parámetros que interesan para comprender el funcionamiento de los sistemas de colas. Estos parámetros son los siguientes:

- *Probabilidad de estado (**p_i**)*: Probabilidad de que un sistema de colas tenga **i** usuarios. Indica también qué porcentaje de tiempo está el sistema en ese estado (cuando i=k el sistema estará lleno).
- *Tasa de llegadas **λ_i**:* Tasa de llegadas cuando el sistema tiene i usuarios dentro antes de que lleguemos. La tasa media será:

$$\lambda = \sum_{i=0}^{K} p_i \cdot \lambda_i$$

- *Tiempo medio de servicio (**μ**)*: Es lo que piden en media los usuarios que acceden al servidor.

- *Tráfico ofrecido (**T_o**)*: Carga de trabajo que la población solicita al sistema. Es posible que dependa del estado del sistema.

$$T_o = \frac{1}{\mu} \sum_{i=0}^{k} p_i \cdot \lambda_i$$

- *Tráfico cursado (**T_c**)*: Parte del tráfico ofrecido que realmente se cursa por el sistema. Se ve con la tasa de salida del sistema.

$$T_c = \frac{1}{\mu} \sum_{i=0}^{k} p_i \cdot \mu_i$$

Podemos verlo también con el número medio de servidores ocupados (m es el número de servidores).

$$T_c = \sum_{i=0}^{m-1} i \cdot p_i + \sum_{i=m}^{k} m \cdot p_i$$

También se considera que es el tráfico ofrecido que no se pierde por culpa de la saturación del sistema.

$$T_c = \frac{1}{\mu} \sum_{i=0}^{k-1} p_i \cdot \lambda_i$$

- *Tráfico perdido (**T_p**)*: Es el que no puede entrar en el sistema porque éste está lleno. El sistema estará en el estado **k**.

$$T_p = T_o - T_c = \frac{1}{\mu} \cdot p_k \cdot \lambda_k$$

- *Tráfico demorado (**T_d**)*: Tráfico que se solicita cuando en el sistema todos los servidores están ocupados.

$$T_d = \frac{1}{\mu} \sum_{i=m}^{k-1} p_i \cdot \lambda_i$$

- *Probabilidad de bloqueo en tiempo*: Es la probabilidad de que el sistema se encuentre lleno.
    - Se considera la cola como recurso (interesa que los usuarios no se pierdan).

$$B_T = p_k$$

    - No se considera la cola recurso (interesa respuesta inmediata).

$$B_T = \sum_{i=m}^{k} p_k$$

- *Probabilidad de bloqueo en llamadas*: Porcentaje de tráfico que se pierde. Define la probabilidad de que el sistema esté lleno (incluida la cola) cuando se produzca una llamada.

$$B_{LL} = \frac{T_p}{T_o} = \frac{p_k \cdot \lambda_k}{\sum_{i=0}^{k} p_i \cdot \lambda_i}$$

- *Probabilidad de demora*: Porcentaje de tráfico cursado que tiene que esperar para ser servido.

$$B_D = \frac{T_d}{T_c} = \frac{\sum_{i=m}^{k-1} p_i \cdot \lambda_i}{\sum_{i=0}^{k-1} p_i \cdot \lambda_i}$$

- *Probabilidad de servicio inmediato*: Probabilidad de que se le dé servicio tan pronto como llega. Ni es rechazado ni tiene que esperar.

$$P_i = 1 - B_D - B_{LL}$$

Se puede comprobar que tanto los tráficos como las probabilidades anteriormente mencionadas están relacionadas de la siguiente forma:

$$T_p = T_o \cdot B_{LL}$$

$$T_d = T_c \cdot B_D$$

$$T_c = T_o \cdot (1 - B_{LL})$$

Otros parámetros de utilidad que sirven para definir y estudiar un sistema de colas son los siguientes:

- *Tasa efectiva de llegadas*: Tasa que realmente entra en el sistema.

$$\bar{\lambda} = \sum_{i=0}^{k-1} \lambda_i \cdot p_i$$

- *Número medio de usuarios en el sistema*:

$$\bar{N} = \sum_{i=0}^{k} i \cdot p_i$$

- *Número medio de usuarios en cola*:

$$\bar{Q} = \sum_{i=m+1}^{k} (i - m) \cdot p_i$$

Con estos últimos parámetros se pueden obtener otros utilizando las "Fórmulas de Little":

- *Tiempo medio de permanencia en el sistema*:

$$\bar{T} = \frac{\bar{N}}{\bar{\lambda}}$$

- *Tiempo medio de espera en cola*:

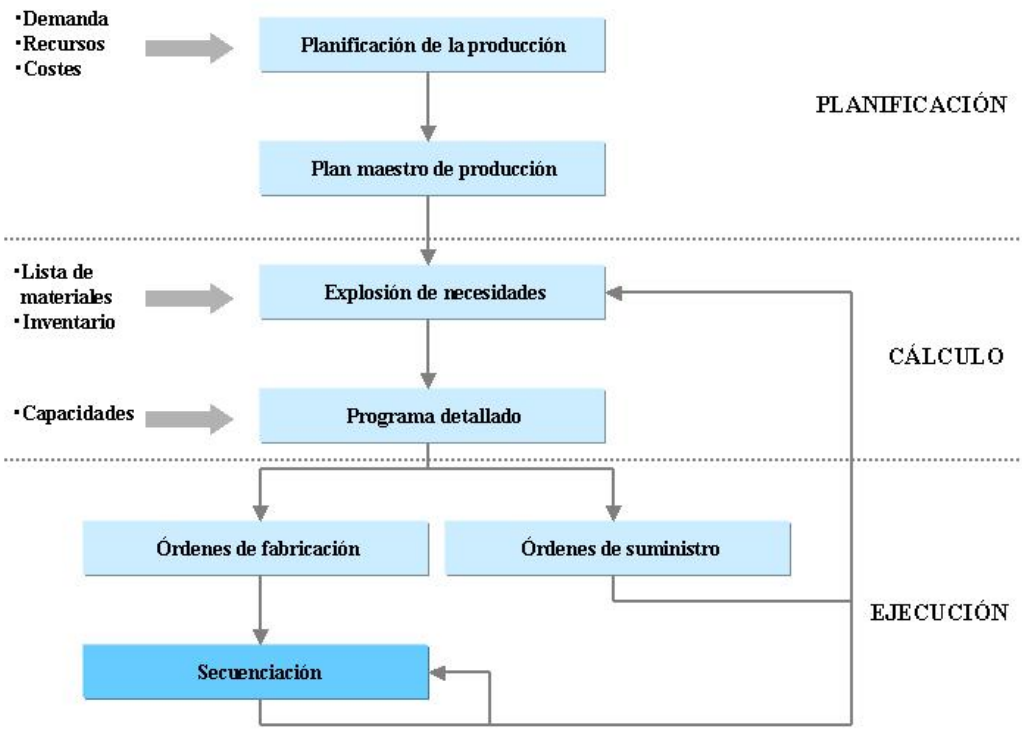
$$\bar{W} = \frac{\bar{Q}}{\bar{\lambda}}$$

Por último se ve un importante parámetro, que es la "*ocupación media de **un** servidor*" o el "*porcentaje de ocupación de **un** servidor*" (**ρ**):

$$\rho = \frac{T_c}{m}$$

Anexo II: Secuenciación y control.

Dentro del esquema jerárquico de gestión de la producción, en el último nivel, se encuentra la fase de ejecución, en la cual se trata de plasmar en la realidad todo lo planificado anteriormente.



En este punto se manejan trabajos concretos, máquinas concretas y fechas de entrega concretas, todo lo cual necesita actuaciones precisas en el día a día de la empresa. En este punto es importante la sincronización de las tareas, así como su ordenación en el tiempo de forma que se realice el trabajo de la forma más eficiente posible para obtener los mejores resultados.

Cuando se realiza la programación de detalle, subyace siempre detrás la política productiva que para ese momento la empresa tenga establecida. Dependiendo de la situación del mercado o de las características propias del taller, existen siempre unos objetivos principales según los cuales se realiza la programación. En unas determinadas circunstancias primará el cumplimiento de las fechas de entrega, mientras en otras podrá ser la minimización de los inventarios intermedios. Pueden citarse las siguientes políticas, cada una de las cuales llevará a un programa diferente en caso de ser elegida:

- *Cumplimiento de las fechas de entrega*. Esta política centra la importancia en la satisfacción del cliente y en asegurar posibles futuros negocios.
- *Minimización del tiempo de las piezas en el proceso*. Esta política trata de reducir los costes de fabricación.
- *Maximizar el uso de los recursos*. Centrando la atención en los cuellos de botella se logrará un aumento en la producción.
- *Mantenimiento de la estabilidad de la planta*. El objetivo podría ser planificar con el objetivo de que haya las menores roturas posibles de los planes realizados.

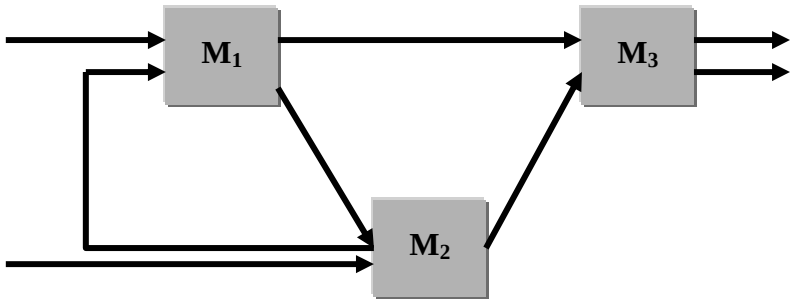
La fase de ejecución, por tanto, es preciso afrontarla tratando de dividir los problemas de acuerdo con la estructura del sistema productivo que se estudia (número de máquinas, complejidad del movimiento de las piezas, tipo de eficiencia buscada), y que dentro de esta fase son varios los pasos que hay que seguir para llegar a la producción de los artículos (asignar máquinas a trabajos, ordenar los trabajos en cada máquina).

La terminología utilizada normalmente en este tipo de problemas (talleres, máquinas, tareas, piezas o trabajos) suele ser específica de determinados entornos de carácter industrial, aunque en realidad los resultados que se obtienen con las técnicas de secuenciación pueden aplicarse en sistemas productivos distintos como hospitales, aeropuertos, etc.

Una clasificación usual de sistemas productivos, en los cuales existe un número *m* de máquinas y una serie de *n* trabajos a procesar, es aquella que centra su atención en el modo en el cual los trabajos fluyen por el taller:

- *Flow-shop (taller serial).* En este caso los trabajos atraviesan el taller siguiendo todos las mismas máquinas y el mismo orden, es decir, la producción es de carácter repetitivo y normalmente continua, de gran volumen y con pocos tipos diferentes de productos finales. Son sistemas muy sensibles a averías en el sentido de que como todos los trabajos atraviesan las mismas máquinas, si cualquiera de ellas deja de funcionar o falta alguna materia prima, se rompe un eslabón que paraliza totalmente el taller.

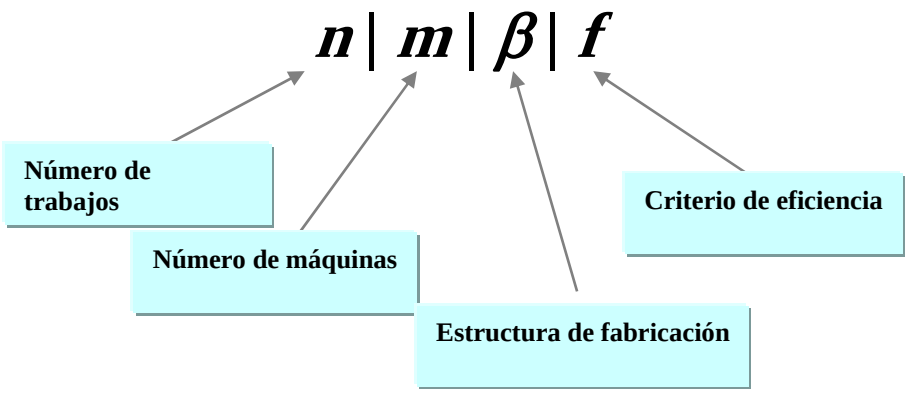
- *Job-shop (taller por trabajos).* Aquí cada trabajo tiene unas características propias que lo diferencia del resto, siguiendo cada uno de ellos una ruta distinta a lo largo del taller. Esta especificidad responde al hecho de que los trabajos se corresponden con órdenes concretas de los clientes (es decir, se produce bajo pedido y con plazos a cumplir), lo cual hace que por lo general se trate de fabricación intermitente.



Dentro del sector industrial podrían identificarse otros tipos de flujos menos frecuentes, tales como la fabricación de productos especiales, cuya característica principal es la singularidad del trabajo. A este tipo de control de la ejecución son aplicables las técnicas de administración de proyectos (PERT).

Una nomenclatura para definir los diferentes tipos de problemas que con estas premisas pueden definirse consiste en identificar los problemas según cuatro parámetros: $<n, m, \beta, f>$:

- *"n"* representa el número de trabajos que se pretende realizar. Los trabajos tienen por nombre $J_1, ..., J_n$.
- *"m"* es el número de máquinas que intervienen en la fabricación de esos trabajos. Pensando en entornos complicados, se puede realizar una clasificación de los tipos posibles de máquinas, los cuales determinarán a su vez las estructuras de fabricación:
    - *Procesadores paralelos*. Todas las máquinas realizan las mismas funciones, y cada trabajo puede ir a cualquiera de ellas. Dependiendo de sus velocidades de procesamiento podrán ser idénticas, uniformes o no relacionadas.
    - *Procesadores dedicados*. Cada operación sólo puede ser realizada en una máquina concreta.
- *"β"* indica la estructura de fabricación que tienen los trabajos a lo largo del taller (tipos de talleres). Los valores usualmente considerados para β son los anteriormente vistos: flow-shop y job-shop.
- *"f"* representa el criterio económico que se trata de cumplir.



Las técnicas de secuenciación y control de la producción pretenden en definitiva proporcionar una planificación horaria de cada operación que compone cada trabajo, de modo que se procesen del modo más satisfactorio según cierto criterio económico. Estos criterios económicos pueden ser utilizados para fijar los objetivos en la búsqueda de soluciones, según los factores que sean más importantes en ese entorno concreto. Hay muchos criterios posibles:

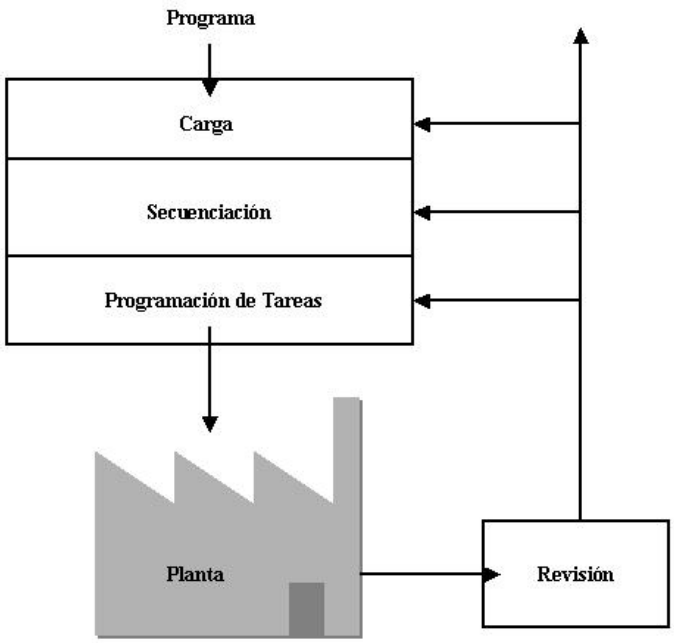
- Criterios basados en los tiempos de finalización de los trabajos. A este tipo pertenece el criterio conocido como *makespan problem*, en el que se trata de buscar la solución que minimiza el tiempo total de producción. Es de interés cuando se desea minimizar la permanencia de material en el sistema.

$$Min\left[C_{\max} = \max\left\{C_i\right\}\right]; \qquad C_i = Tiempo\ en\ el\ que\ J_i\ es\ \ finalizado.$$

- Criterios basados en los tiempos contractuales de finalización. Representamos por d_i la fecha en que hemos comprometido con el cliente o que ha sido impuesta por consideraciones internas de planificación la entrega del trabajo J_i. Definimos entonces el huelgo del trabajo, L_i, como la diferencia entre la fecha de finalización y la fecha comprometida. Dos criterios económicos asociados son la minimización del valor máximo de L_i, o bien de la suma de todos los L_i, siempre con el compromiso de satisfacer las demandas de los clientes.

$$\min\left[L_{\max} = Max\left\{L_i\right\}\right];$$

En el caso más general, las diversas fases que componen este último nivel de la gestión de la producción son las siguientes:



- *Carga de las máquinas*. Tenemos por una parte trabajos a procesar y por otra las máquinas necesarias para su procesamiento. En muchas ocasiones más de una máquina puede procesar el mismo trabajo, siendo el objeto de esta primera fase la distribución de los trabajos entre las diferentes máquinas, atendiendo principalmente a los costes de cada una, sus capacidades y las características del personal que las atiende.
- *Secuenciación de los trabajos*. Una vez se conocen los trabajos que cada centro tiene que realizar, se debe proceder a establecer el orden en que los trabajos van a ser procesados. Se atenderá básicamente a la prioridad de cada orden, las fechas de entrega comprometidas y al tiempo de procesamiento estándar de cada uno.

- *Programación de tareas*. En determinados entornos productivos es posible establecer unos objetivos horarios (**scheduling**) en función de la secuenciación realizada y de los tiempos estándar, que sirve de guión para analizar si la ejecución se realiza conforme a lo previsto.
- *Revisión de la ejecución*. Con cierta periodicidad es necesario seguir la evolución de la producción en el taller con el fin de observar si se van cumpliendo los planes establecidos o bien es necesario establecer alguna medida correctora (resecuenciación).

Anexo III: Resumen de Software Moderno de Simulación.

A continuación se presenta una tabla con los principales paquetes de software de simulación que se puede encontrar en el mercado en la actualidad (revisión de 1999).

Producto y Empresa	Aplicaciones típicas
ALPHA/Sim ALPHATECH Inc.	Ingeniería de procesos económicos, distribución de servicios, sistemas de producción, sistemas de control militar,...
AMS: Semiconductor Manugistics Inc.	Utilizado en la industria de semiconductores para planificación de líneas de producción de componentes.
Arena Systems Modeling Corporation	Análisis de procesos, planificación, simulación de eventos discretos, modelado de sistemas,...
@RISK PALISADE Corporation	Simulación de propósito general, análisis de costes, temas financieros.
AutoMod AutoSimulations	Fabricación, almacenamiento y distribución, definición de células de trabajo, secuencias de compras,...
C Library Numerical Algorithms Group	Aplicaciones generales en el campo de las finanzas, industria, universidad, investigación,...
Call Center MAESTRO The Model Builders Inc.	Diseño de Call Centres y evaluación tecnológica.
CB Predictor Decisioneering Inc.	Control de inventarios, previsión de ventas, ...
COMNET III CACI Products Company	Comunicación de datos, telecomunicaciones, diseño y planificación de redes,...
Cristal Ball Pro Decisioneering Inc.	Análisis y planificación de negocios, gestión del riesgo, análisis de coste / beneficio,...
DecisiónPro 3.0 Vanguard Software Corporation	Modelado de sistemas económicos y financieros, modelos de decisión.
Deneb/Quest Deneb Robotics Inc.	Procesos de fabricación, gestión de materiales, análisis de flujos discretos de materiales,...
ExpertFit Averill M. Law & Associates	Ajuste de distribuciones de probabilidad a datos.
Extend Imagine That Inc.	Diseño de ingeniería, análisis científico,...
Extend & BPR Imagine That Inc.	Modelado de procesos, análisis estratégico, flujos de trabajo, análisis de inserción de tecnologías,...

Extend & BPR & Manufacturing Imagine That Inc.	Simulación general, teoría de colas, producción, análisis de costes, modelado industrial y comercial,...
Extend & Manufacturing Imagine That Inc.	Modelado industrial y comercial, teoría de colas, logística, análisis de costes,...
FACTOR/AIM SYMIX/Pritsker Division	Soporte de decisiones de producción.
GPSS/H Wolverine Software Corporation	Simulación de sistemas en general, producción, transporte, redes informáticas, logística, teoría de colas,...
GPSS/PC Minuteman Software	Simulación para optimización o análisis de diseños en general.
H/SCHED Haverly Sistemas Inc.	Simulación de procesos específicos de la industria petroquímica.
IBM Suply Chain Simulator IBM	Análisis de cadenas de suministro, análisis de costes, optimización de inventarios,...
Ithink Analyst 5.1.1 High Perfomance Sistems Inc.	Modelado de estrategias de alto nivel. No dispone de funcionalidad discreta.
JobTime Plus JobTime Systems Inc.	Planificación avanzada basada en simulación, teoría de colas,...
MAST Simulation Environment CMS Research Inc.	Sistemas de producción.
MedModel PROMODEL Corporation	Optimización y diseño de infraestructuras, análisis de capacidad,..Especialmente indicado para entornos de medicina.
Micro-GPSS Ingolf Stahl	Simulación de eventos discretos en general.
Micro Saint Micro Analysis & Design Inc.	Sistemas de producción, estudios del factor humano, simulación en general.
MODSIM III CACI Products Company	Problemas de transporte, telecomunicaciones, tráfico, logística,...
NAG Libraries Numerical Algorithms Group	Problemas de simulación en general.
Optima 2.5 Micrografx Inc.	Análisis de procesos de telecomunicación, financieros, de producción,...
PASION32 Stanislaw Raczynski	Simulación de sistemas de producción, investigación operativa,...
ProcessModel PROMODEL Corporation	Simulación en general.
ProModel PROMODEL Corporation	Procesos de producción, diseño de infraestructuras, optimización de procesos,...

Proof Animation Wolverine Software Corporation	Procesos de producción, Manejo de materiales, transporte,...
PS-ENGINE Prosolvia AB	Fabricación de procesos discretos.
QueGauss Aptech Systems Inc.	Simulación de teoría de colas.
ServiceModel PROMODEL Corporation	Simulación en general, teoría de colas, estudio de costes, optimización,...
SIGMA Custom Simulations	Simulación de eventos discretos, procesos productivos, modelado de equipos,...
Silk ThradTec Inc.	Procesos productivos, comunicaciones.
SimGauss Aptech Systems Inc.	Simulación dinámica de ecuaciones diferenciales.
SIMNON 3.0 SSPA Maritime Consulting AB	Ecuaciones diferenciales y ecuaciones en diferencias, simulación de sistemas dinámicos.
SIMPLE++ Tecnomatix Technologies Inc.	Simulación, planificación de la producción, programación orientada a objetos.
SIMPROCESS CACI Products Company	Procesos económicos.
SIMUL8 Visual Thinking International	Herramienta visual para la simulación de eventos discretos.
SLX Wolverine Software Corporation	Procesos de producción, manejo de materiales, transporte,...
Stat:Fit Geer Mountain Software Corp.	Ajuste de distribuciones para aplicaciones de simulación.
Taylor ED Logistics Suite F&H Simulations	Procesos de producción, manejo de materiales.
Visual Simulation Environment Orca Computer Inc.	Simulación en general.
WITNESS 9.0 Lanner Group Inc.	Ingeniería de procesos, teoría de colas, análisis de secuencias, análisis de costes,...
WorkFlow Analyzer Meta Software Corporation	Ingeniería de procesos, workflow,...