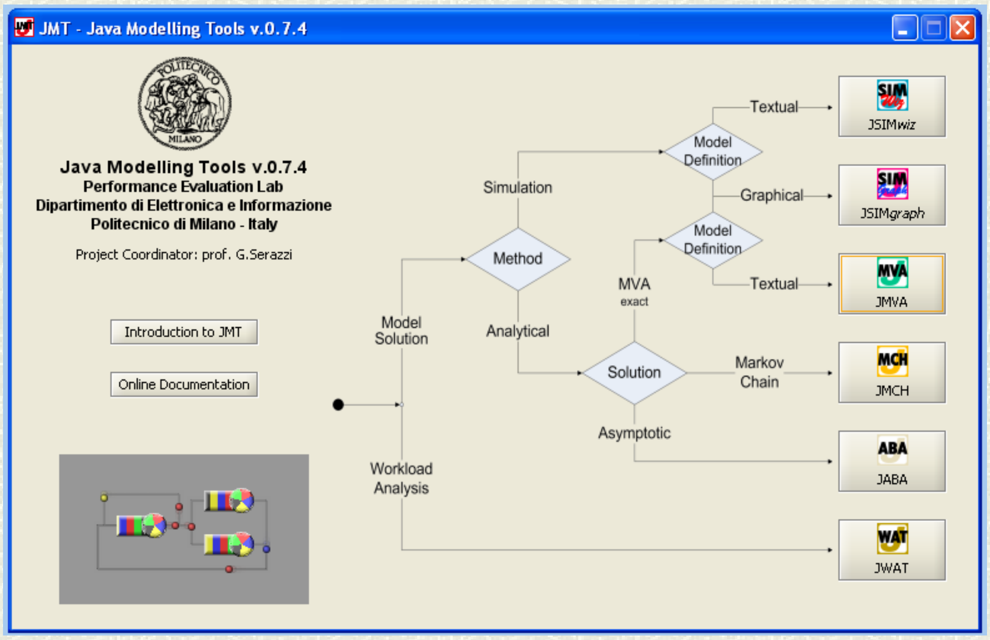
Diseño y Evaluación de Configuraciones

Java Modelling Tools(JMT): JMVA

J.M. Drake

Notas:

Java Modelling Tools (Politécnico de Milán)



Notas:

The Java Modelling Tools (JMT) is a free open source suite consisting of six tools for performance evaluation, capacity planning, workload characterization, and modelling of computer and communication systems. The suite implements several state-of-the-art algorithms for the exact, asymptotic and simulative analysis of queueing network models, either with or without product-form solution. Models can be described either through wizard dialogs or with a graphical user-friendly interface. The workload analysis tool is based on clustering techniques. The suite incorporates an XML data layer that enables full reusability of the computational engines.

JMVA: for the exact analysis of single-class or multiclass product-form queueing networks, processing open, closed or mixed workloads. A stabilized version of the Mean Value Analysis MVA algorithm is used. Network structure is specified by textual wizards. What-if analyses and graphical representation of the results are provided.

JABA: for the identification of bottlenecks in multiclass closed product-form networks using efficient convex hull algorithms. Up to three customer classes are supported. It is possible to identify potential bottlenecks corresponding to the different mixes of customer classes in execution. Optimization studies (e.g., throughput maximization, minimization of response time, identification of the optimal load) can be performed through the identification of the saturation sectors, i.e., the mixes of customer classes in execution that saturate more than one resource simultaneously.

JSIMwiz and **JSIMgraph**: JSIM is a discrete-event simulator of queueing network models. JSIMwiz is a wizard-based interface for the discrete-event simulator. A sequence of wizard windows helps in the definition of the network properties. The JSIMgraph is a graphical user-friendly interface for the same simulator. The simulation engine supports several probability distributions for characterizing service and inter-arrival times, load-dependent strategies using arbitrary functions of the current queue-length can be specified. They support state-independent routing strategies, e.g., Markovian or round robin, as well as state-dependent strategies, e.g., routing to the server with minimum utilization, or with the shortest response time, or with minimum queue length. The simulation engine supports several extended features not allowed in product-form models, namely, finite capacity regions (i.e., blocking), fork-join servers (i.e., parallelism), and priority classes.

JMCH: it applies a simulation technique to solve a single station model, with finite (M/M/1/k) or infinite queue (M/M/1), and shows the underlying Markov Chain.

JWAT: supports the workload characterization process. Some standard formats for input file are provided (e.g., Apache HTTP and IIS log files), customized formats may also be specified. The imported data can initially be analyzed using descriptive statistical techniques (e.g, means, correlations, histograms, boxplots, scatterplots), either for univariate or multivariate data. Algorithms for data scaling, sample extraction, outlier filtering, k-means and fuzzy k-means clustering for identifying similarities in the input data are provided. These techniques allow the identification of cluster of customers having similar characteristics. The clusters centroids represent the mean values of the parameters of the classes that can be used for the workload parameterization.

JMVA: Exact MVA algorithm

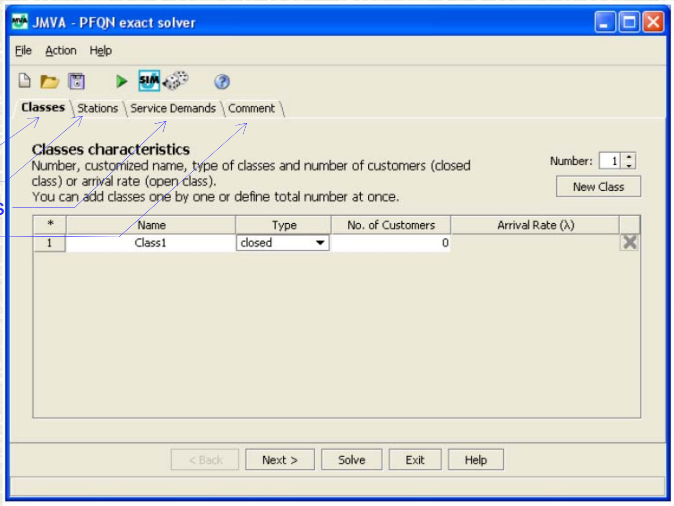
- Permite estudiar aplicando los algorithm MVA sistemas abiertos, sistemas cerrados y sistemas híbridos, con una clase y múltiples clases
- Admite colas con:
  - Colas con tiempo de servicio independiente de la carga.
  - Colas con tiempos de servicio dependientes de la carga.
  - Delays
- Permite evaluar :
  - Throughput
  - Queue length
  - Residence times
  - Utilizations
  - De cada cola
  - Del sistema completo
- Análisis con barrido de los parámetros (What if) y salida gráficas.
- Proporciona una GUI para la introducción de los datos y mostrar los resultados.

Notas:

JMVA: Interfaz gráfica de usuario

- El proceso de análisis de una red de colas es:
  - Definición del modelo.
  - Resolución del modelo
  - Presentación de los resultados

Pestañas:
-Define los clientes
-Define las colas
-Parametros de las colas
-Comentarios

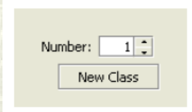


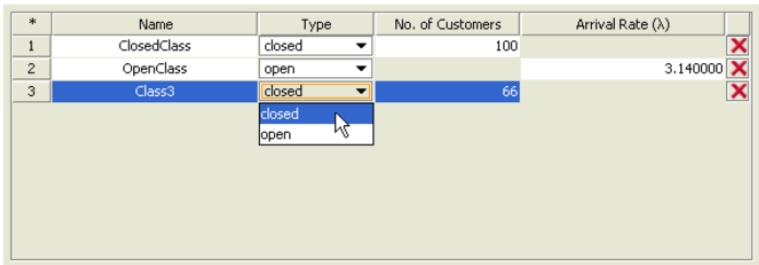
Menú
Toolbar
Pestañas de datos
Botones de control

Notas:

Pestaña Classes

- Permite definir los tipos de clientes:
  - Open: Definidos por su tasa de llegada
  - Closed: Definidos por su número



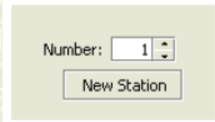


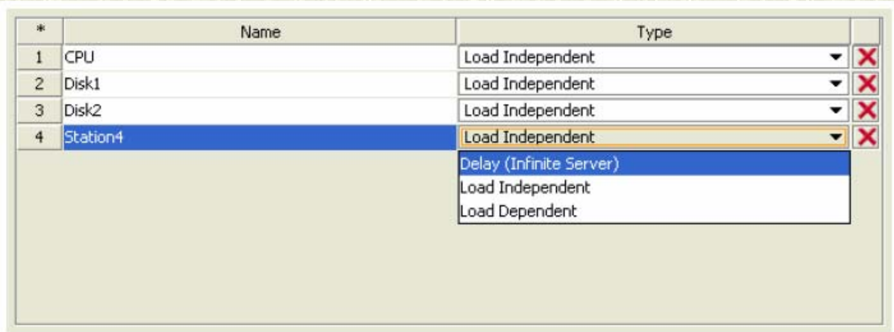
Notas:

Pestaña de definición de las colas

- Permite definir el número y la naturaleza de las colas que forman el modelo.
- Los tipos de colas pueden ser:
  - Independientes de la carga (normales: su tiempo de servicio no es función de la carga)
  - Dependiente de la carga: Su tiempo de servicio es función del nº de cliente en la cola)*
  - Retraso: Los clientes no se encolan (thinking time)

(*) Se puede utilizar colas dependientes de la cargas sólo si hay una única clase de clientes cerrada.





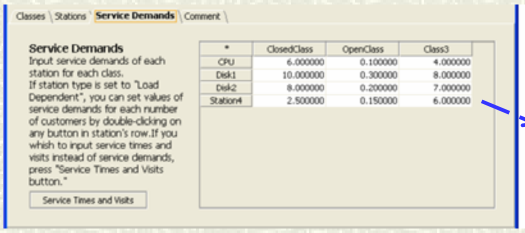
Notas:

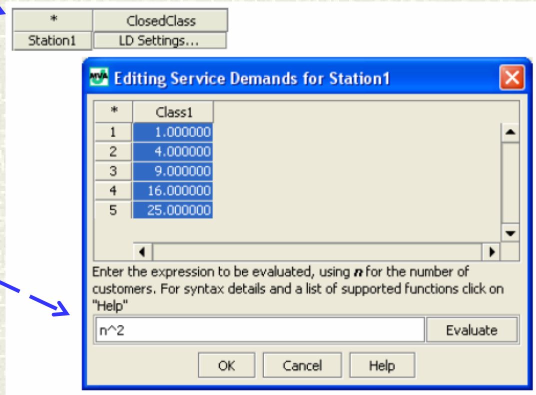
Pestañas Demanda de servicios, tiempo de servicio y tasa de visitas

- Ofrece dos opciones:
  - A través de las demandas de servicios (D_{kc})
  - Combinando Tiempos de servicio y tasas de visitas

$$\left.\begin{matrix} V_{kc} \\ S_{kc} \end{matrix}\right| \Rightarrow D_{kc} = V_{kc} \times S_{Kc}$$

$$D_{kc} \Big| \Rightarrow \begin{matrix} V_{kc} = \begin{cases} 1 \text{ if } D_{kc} > 0 \\ 0 \text{ if } D_{kc} = 0 \end{cases} \\ Skc = \begin{cases} D_{kc} \text{ if } D_{kc} > 0 \\ 0 \text{ if } D_{kc} = 0 \end{cases} \end{matrix}$$





*JFEP*³ (Java Fast Expression Parser)

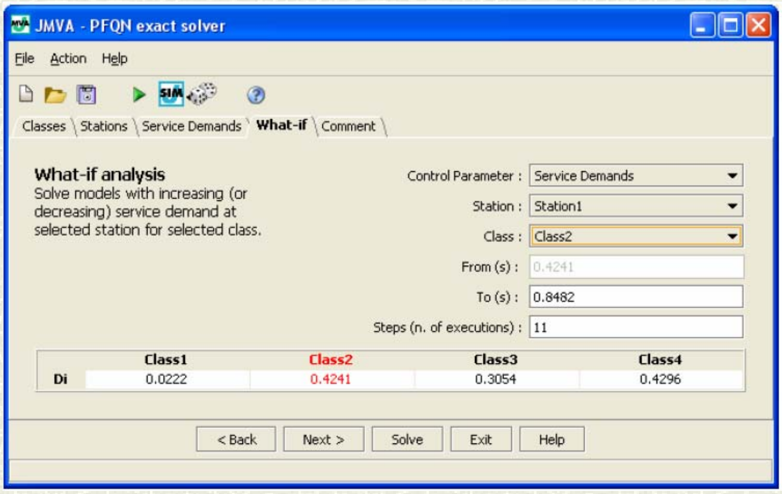
Operator	Symbol
Power	^
Unary Plus, Unary Minus	$+n$, $-n$
Modulus	%
Division	/
Multiplication	*
Addition, Subtraction	+, −

Function	Symbol
Sine	sin()
Cosine	cos()
Tangent	tan()
Arc Sine	asin()
Arc Cosine	acos()
Arc Tangent	atan()
Natural Logarithm	ln()
Logarithm base 10	log()
Absolute Value / Magnitude	abs()
Random number [0, 1]	rand()
Square Root	sqrt()
Sum	sum()

Notas:

Pestaña What-if

- Se utiliza para realizar estudios repetidos de un odelo bajo el cambio de un parámetro.
- Parámetros de control:
  - Disable
  - Número de clientes
  - Tasa de llegada
  - Population mix
  - Demanda de servicio
- El parámetro elegido se selecciona el rango de variacíon y el número de experimentos



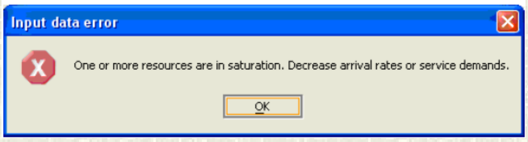
Notas:

Solución del modelo

- El modelo se resuelve pulsando el botón "solve"
- En el caso de que el modelo no sea estable:

$$\max_k \sum_c \lambda_c \times D_{kc} < 1$$

Genera un aviso de que no puede ser resuelto.

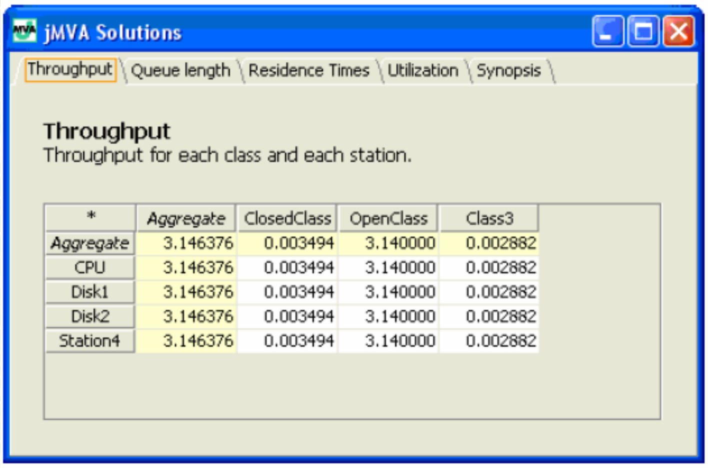


- Resuelto el modelo genera una nueva ventana de resultados que puede ser tabular o gráfica si hay definidos What-if

Notas:

Solución del modelo: Presentación en tabla

- En la tabla se proporciona los resultados de cada métrica organizada por clase y agregadas por clases y recurso.

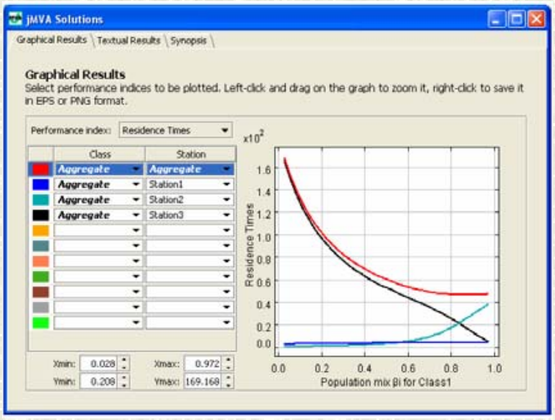


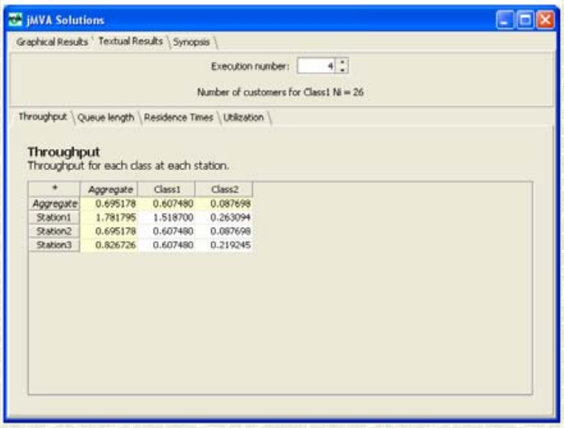
*	Aggregate	ClosedClass	OpenClass	Class3
Aggregate	3.146376	0.003494	3.140000	0.002882
CPU	3.146376	0.003494	3.140000	0.002882
Disk1	3.146376	0.003494	3.140000	0.002882
Disk2	3.146376	0.003494	3.140000	0.002882
Station4	3.146376	0.003494	3.140000	0.002882

Notas:

De la salida de unasolución con What-if

- Proporciona información gráfica, tabular y un breve resumen del análisis.

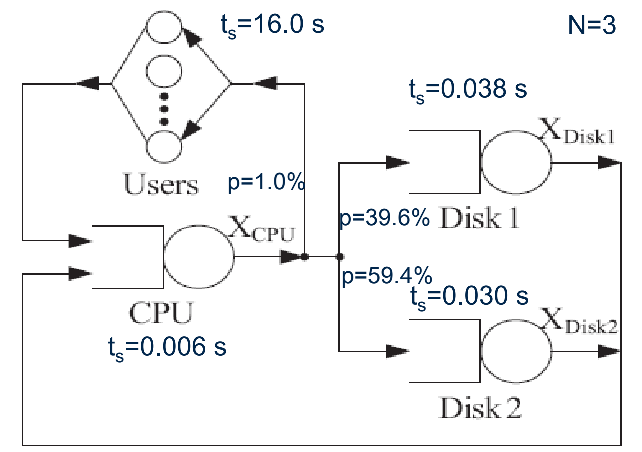




*	Aggregate	Class1	Class2
Aggregate	0.695178	0.607480	0.087698
Station1	1.781795	1.518700	0.263094
Station2	0.695178	0.607480	0.087698
Station3	0.826726	0.607480	0.219245

Notas:

Caso estudio 1: Sistema cerrado



Analizar el sistema para N=3

Analizar el throughput del sistema cuando N varía de 1 a 10

Notas:

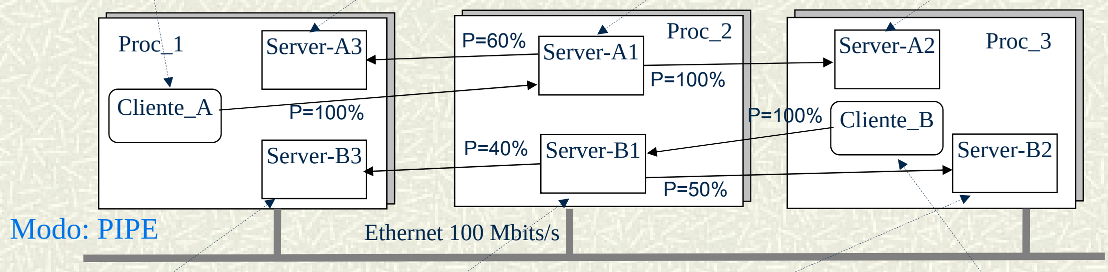
Caso estudio 2: Sistema abierto de colas

Cliente_A: (Host: Proc_1)
Gen: Const (T_P=0.2 s)
Load: Exp (T_L=0.05 s)
Mens: Const (L_M=256 B)
Out: (Proc_2, port:12000)

Server A3: (Host: Proc_2)
Input: (Port:12002)
Load: Exp (T_L=0.03 s)
Out1:No conectada
Out2: No conectada

Server A1: (Host: Proc_3)
Input: (Port:12000)
Load: Exp (T_L=0.05 s)
Out1:UDP-IP(Proc_1,12002)
Out2:UDP-IP(Proc_3,12004)

Server A2: (Host:Proc_3)
Input: (Port:12004)
Load: Exp (T_L=0.03 s)
Out1:No conectada
Out2:No conectada



Server B3: (Host: Proc_1)
Input: (Port:12010)
Load: Unif(T_L=0.55s,Rang=0.15)
Out1:No conectada
Out2: No conectada

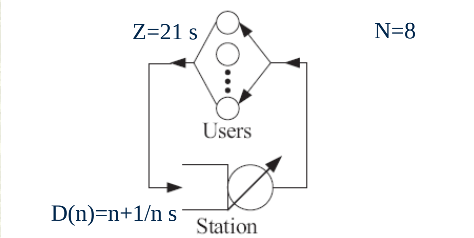
Server B1: (Host: Proc_2)
Input:(Port:12000)
Load:Unif (T_L=0.5 s,Rang=0.2s)
Out1:UDP-IP(Proc_3,12008)
Out2:UDP-IP(Proc_1,12010)

Server B2: (Host: Proc_3)
Input:(Port:12008)
Load:Unif(T_L=0.45s,Rang=0.1s)
Out1:No conectada
Out2:No conectada

Cliente_B: (Host: Proc_3)
Gen: Const (T_P=2.02s)
Load: Exp (T_L=0.5s)
Mens: Const (L_M=25 KB)
Out: (Proc_2, port:12006)

Notas:

Caso de estudio 3: Clase con demanda función de la carga



n	1	2	3	4	5	6	7	8
$D(n)$ [s]	2.00	2.50	3.33	4.25	5.20	6.17	7.14	8.13

Notas: