Performance Evaluation Lab
Dipartimento di Elettronica e Informazione
Politecnico di Milano - Italy

Java Modelling Tools

users manual

Version 0.3, June 30, 2006

Contents

Chapter 1

Introduction

The Java Modelling Tools (JMT) is a free open source suite for performance evaluation, capacity planning and modelling of computer and communication systems. The suite implements numerous 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 suite includes also a workload analysis tool based on clustering techniques. The suite incorporates an XML data layer that enables full reusability of the computational engines.

The JMT suite is composed of *six* tools that support different analyses frequently used in capacity planning studies. The main features of each tool follows.

JSIM: a discrete-event simulator for the analysis of queueing network models. An intuitive sequence of *wizard* windows helps specifying network properties. 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. JSIM supports 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, and priority classes. The analysis of simulation results employs on-line transient detection techniques based on spectral analysis. What-if analyses, where a sequence of simulations is run for different values of parameters, are also possible.

JMODEL: a graphical user-friendly interface for the simulator engine used by JSIM. It integrates the same functionalities of JSIM with an intuitive graphical workspace. This allows an easy description of network structure, as well as a simplified definition of the execution features like blocking regions. Network topologies can be exported in vectorial or raster image formats.

JMVA: meant for the exact analysis of single or multiclass product-form queueing network models, either processing open, closed or mixed workloads. The classic MVA solution algorithm is used. Network structure is specified by textual *wizards*, with conversion functions from probabilities to average visit ratios (and viceversa). What-if analyses are allowed.

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. It is also possible to dynamically change the arrival rate and service time of the system.

JABA: a tool for the identification of bottlenecks in closed product-form networks using efficient convex hull algorithms. The tool supports models with up to three job classes. It is possible to identify potential bottlenecks corresponding to the different mixes of customer classes. Models with thousands of queues can be analyzed efficiently. The saturation sectors, i.e., the mixes of customer classes that saturate more than one resource simultaneously, are identified.

JWAT: supports the workload characterization phase, with emphasis on Web log data. Some standard formats for input file are provided (e.g., Apache HTTP log files), and customized formats may also be specified. The imported data can initially be analyzed using descriptive statistical techniques (e.g, means, correlations, pdf 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 to determine cluster centroids, and then estimate mean workload and service demands to be used for model parametrization. The tool includes also an interface to the similarity clustering tool CLUTO.

1.1 Starting the JMT suite

Double click on the JMT icon on your *program group* or *desktop*, or open the *command prompt* and type from the installation directory:

```
java -jar JMT.jar
```

the window of Figure 1.1 will be shown.

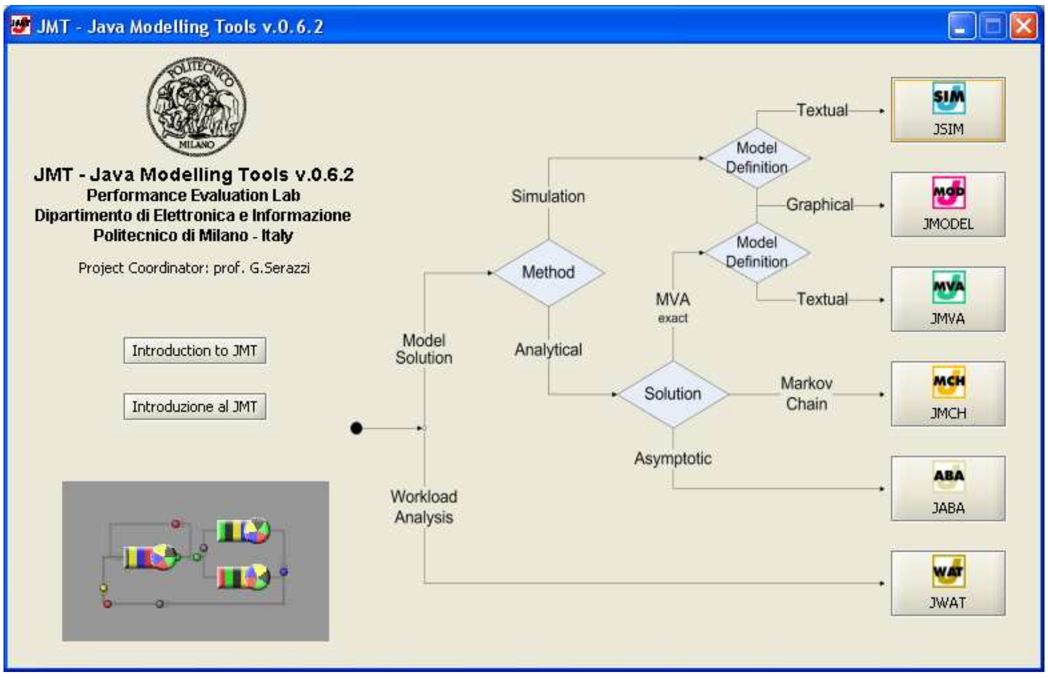


Figure 1.1: JMT suite Starting Screen

This starting screen is used to select the application of the suite to be executed by clicking on the corresponding button. The flow chart try to help the user to chose the application that best fits its needs.

In the following chapters all of the tools[1] will be examined with details and examples. This manual is intended for the general user that wants to learn how to interact with Java Modelling Tools; advanced users that want to learn details on internal data structures, computational engines and XML interfaces should refer to *JMT system manual*.

[1]for the moment only *JMVA*

Chapter 2

JMVA

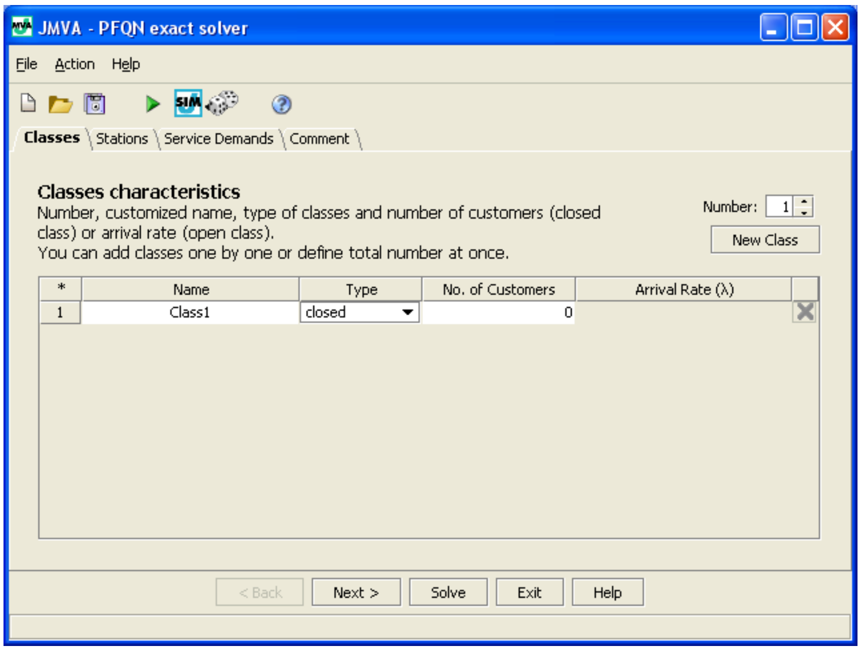
2.1 Overview

JMVA solves open, closed and mixed product form [BCMP75] queueing networks with the exact MVA algorithm [RL80]. In order to avoid fluctuations of the solutions when the model contains load dependent stations, the implemented algorithm is a stabilized version [BBC$^+$81] of the classic MVA algorithm.

Resources may be of two types: *queueing* (either with `load independent` or `load dependent` service times) and `delay`. The model is described in alphanumeric way: user is guided through the definition process by steps of a *wizard* interface (5 or 6 steps). What-if analyses, where a sequence of model are solved for different values of parameters, are also possible (see subsection 2.2.5). A graphical interface to describe the model in a user-friendly environment is also available, see **??** for details.

2.1.1 Starting the alphanumeric MVA solver

Selecting button on the starting screen, Figure 2.1 window shows up. Three main areas are shown:



Classes characteristics
Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class).
You can add classes one by one or define total number at once.

Number: 1

*	Name	Type	No. of Customers	Arrival Rate (λ)
1	Class1	closed	0	

Figure 2.1: Classes tab

Menu : it is organized into three groups of functions. To use a menu, click on the menu heading and choose the appropriate option. For the description of menu entries, see section 2.3

Toolbar : contains some buttons to speed up access to JMVA functions (e.g. New model, Open, Save. . . See section 2.3 for details). If you move the mouse pointer over a button a tooltip will be shown up.

Page Area : this is the core of the window. All MVA parameters are grouped in different tabs. You can modify only a subset of them by selecting the right tab, as will be shown later.

2.2 Model definition

Models with one or multiple customer classes provide estimates of performance measures. For a brief description of basic terminology please refer to Appendix A.

In the case of single class models, the workload is characterized by two inputs: the set of service demands, one for each resource, and the workload intensity. On the other hand, in multiple class models, a matrix of service demands is requested [LZGS84].

2.2.1 Defining a new model

To define a new model select toolbar button or the *New* command from *File* menu. The following parameters must be defined:

1. `Classes` with their workload intensities (number of customer N for closed classes and arrival rate λ for open classes)
2. `Stations` (service centers)
3. `Service demands` (or Service Times and Visits)
4. Optional short `Comment`

The execution of JMVA provides, for each class and each station, the following performance indices:

- Throughput
- Queue Length
- Residence Time
- Utilization

The following *aggregate* indices are provided:

- System Throughput
- System Response Time
- Average number of customers in the system

Input tabs

As can be seen in Figure 2.1, the parameters that must be entered in order to define a new model are divided in four tabs: `Classes`, `Stations`, `Service Demands` and `Comment`.

Tabs number can become five, if you click *Service Times and Visits* button in `Service Demands Tab`. As will be discussed in subsection 2.2.4, the `Service Demands Tab` will be hidden and it will appear `Service Times Tab` and `Visit Tab`. You can navigate through tabs:

- using sequential wizard buttons, if enabled, at the bottom of the window (Figure 2.2)
- using sequential buttons located in menu
- using the tab selector, clicking on the corresponding tab (Figure 2.3)

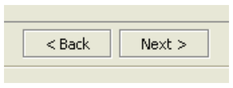


Figure 2.2: Wizard buttons

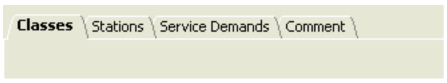


Figure 2.3: Tab selector

2.2.2 Classes Tab

An example screenshot of this tab can be seen in Figure 2.1. This tab allows to characterize customer classes of the model. Your model will be a single class model if and only if there will be only one class, closed or open. On the contrary multiple class models will have at least two classes, closed and/or open.

The number of classes in the model can be specified in the corresponding input area, shown in Figure 2.4 and can be modified using the keyboard or using the spin controls.

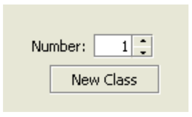
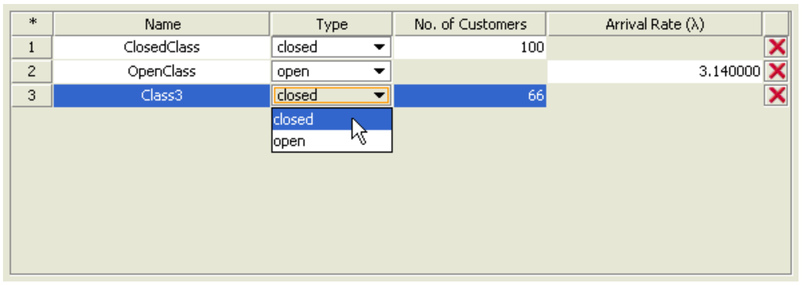


Figure 2.4: Number of classes

Using the delete button ❌ associated to a specific class, a class can be removed provided that there will be at least one class after the deletion. Similar result may be obtained using spin controls, decreasing classes number; in this case last inserted class will be removed.

Default class names are *Class1*, *Class2*, ... *ClassN*. A model can be personalized by changing this names.

In Figure 2.5 there are three classes of customers, two closed and one open. The third class has the default name *Class3* while the other two classes have personalized names, namely *ClosedClass* and *OpenClass*.

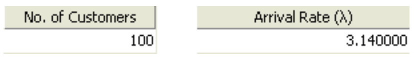


*	Name	Type	No. of Customers	Arrival Rate (λ)	
1	ClosedClass	closed	100		❌
2	OpenClass	open		3.140000	❌
3	Class3	closed	66		❌

Figure 2.5: Defining the classes types

A class type can be `Open` or `Closed`. It is important to define each class type because a closed class workload is described by the number of customers in each class and the open classes workload is described by the customer arrival rate for each class.

As can be seen in Figure 2.5, a class type can be selected in a combo-box. The input boxes *No. of Customers (N)* referring to closed classes accept only positive integer numbers; the input boxes of the *Arrival Rate* (λ) referring to open classes, accept positive real numbers (Figure 2.6).



No. of Customers	Arrival Rate (λ)
100	3.140000

Figure 2.6: Workload definition of the number of customers of a closed class ($N = 100$) and the arrival rate of an open class ($\lambda = 3.14$)

2.2.3 Stations Tab

The number of stations of the model can be specified in the corresponding input area (Figure 2.7) and can be modified using the keyboard or the spin controls.

Using the delete button ❌ associated to a specific station, a station can be removed provided that there will be at least one station after the deletion. Similar result may be obtained using spin controls, decreasing stations number; in this case last inserted station will be removed.

Default station names are *Station1*, *Station2*, ... *StationN*. In order to personalize your model, you can change and give names other than default ones.

In Figure 2.8 there is only one station with default name *Station4* and there are three stations with personalized names: *CPU*, *Disk1* and *Disk2*.

A station type can be `Load Independent`, `Load Dependent` or `Delay`. You can insert in your model a `Load Depend` center only if there is a unique closed class[1]; in all other cases the combo-box will be disabled.

[1]Multiclass, open and mixed models with load dependent stations are not supported yet

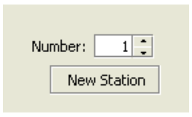
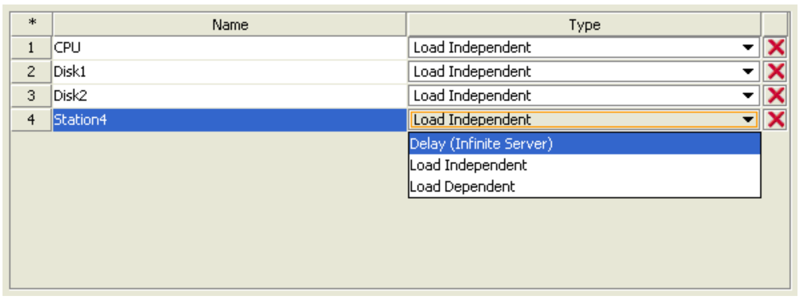


Figure 2.7: Number of stations



*	Name	Type
1	CPU	Load Independent
2	Disk1	Load Independent
3	Disk2	Load Independent
4	Station4	Load Independent

Figure 2.8: Defining the stations type

It is important to define each station type because if a station is `Load Dependent` a set of service demand - or a set of service times and the number of visit - must be defined (one service demand/time for each possible value of queue length inside the station).

In subsection 2.2.4 we will explain this concept with more details.

2.2.4 Service Demands, Service Times and Visits Tabs

Service Demands can be defined in two ways:

- directly, by entering Service Demands (D_{kc})
- indirectly, by entering Service Times (S_{kc}) and Visits (V_{kc})

Service demand D_{kc} is the total service requirement, that is the average amount of time that a customer of class c spends in service at station k during one interaction with the system, i.e. it's complete execution. Service time S_{kc} is the average time spent by a customer of class c at station k for a single visit at that station while V_{kc} is the average number of visits at that resource for each interaction with the system.

Remember that $D_{kc} = V_{kc} * S_{kc}$ so it's simple to compute service demands matrix starting from service times and visits matrixes. Inverse calculation is performed with the following algorithm:

$$V_{kj} = \begin{cases} 1 & \text{if} \quad D_{kc} > 0 \\ 0 & \text{if} \quad D_{kc} = 0 \end{cases}$$

$$S_{kc} = \begin{cases} D_{kc} & \text{if} \quad D_{kc} > 0 \\ 0 & \text{if} \quad D_{kc} = 0 \end{cases}$$

Service Demands Tab

In this tab you can insert directly Service Demands D_{kc} for each pair {station k-class c} in the model. In Figure 2.9 a reference screenshot can be seen: notice that a value for every D_{kc} element of the D-matrix has already been specified because default value assigned to newly created stations is zero.

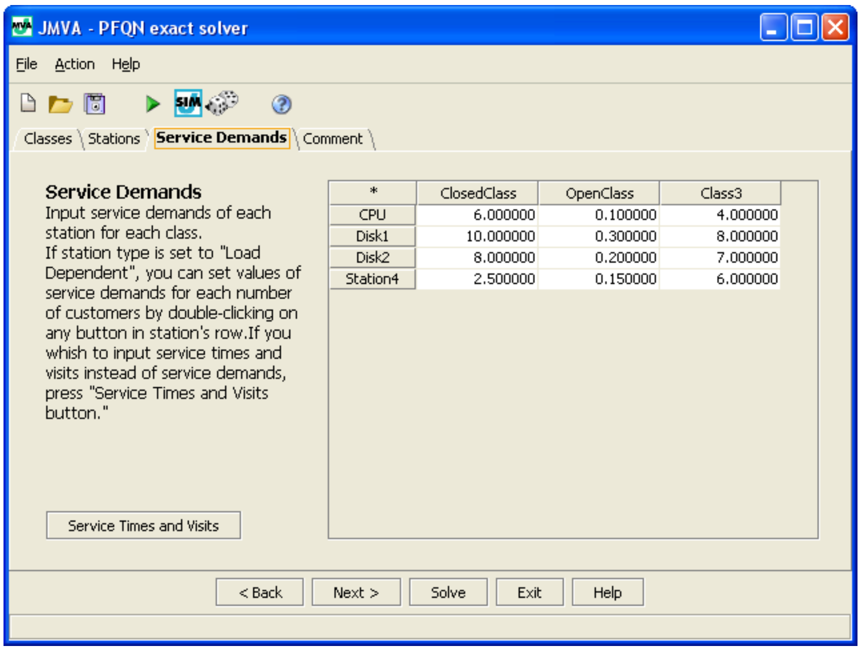
In the example of Figure 2.9, each job of type *ClosedClass* requires an average service demand time of 6 sec to *CPU*, 10 sec to *Disk1*, 8 sec to *Disk2* and 2.5 sec to *Station4*.
On the other hand, a job of type *OpenClass* requires on average 0.1 sec of *CPU* time, 0.3 sec of *Disk1* time, 0.2 sec of *Disk2* time and 0.15 sec of *Station4* time to be processed by the system.

If the model contains any load dependent station, the behavior of Figure 2.10 will be shown.

By double-clicking on *LD Settings...* button a window will show up and that can be used to insert the values of the service demands for each possible number of customer inside the station. That values can be computed by evaluating an analytic function as shown in Figure 2.11. The list of supported operators and more details are reported in subsection 2.2.7.

Service Times and Visits Tabs

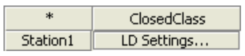
In the former tab you can insert the Service Times S_{kc} for each pair {station k-class c} in the model, in the latter you can enter the visits number V_{kc} (See Figure 2.12).



Service Demands
Input service demands of each station for each class.
If station type is set to "Load Dependent", you can set values of service demands for each number of customers by double-clicking on any button in station's row.If you whish to input service times and visits instead of service demands, press "Service Times and Visits button."

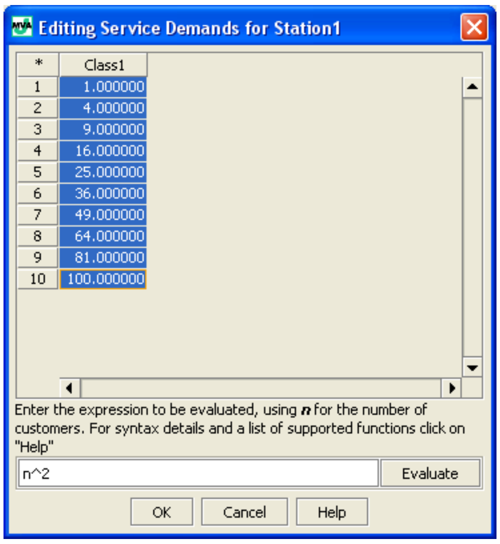
*	ClosedClass	OpenClass	Class3
CPU	6.000000	0.100000	4.000000
Disk1	10.000000	0.300000	8.000000
Disk2	8.000000	0.200000	7.000000
Station4	2.500000	0.150000	6.000000

Figure 2.9: The Service Demands Tab



*	ClosedClass
Station1	LD Settings...

Figure 2.10: Defining a *load dependent* station service demand

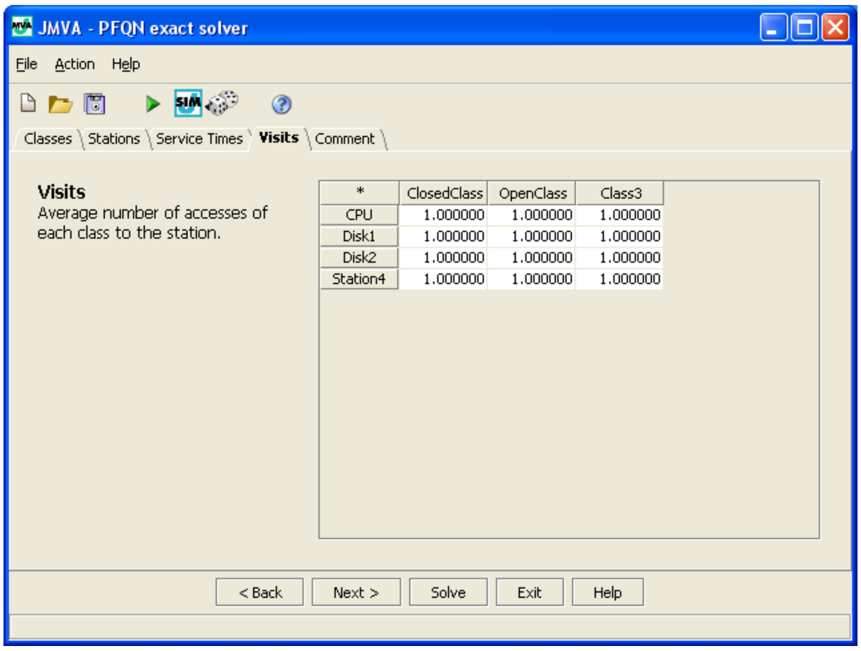


*	Class1
1	1.000000
2	4.000000
3	9.000000
4	16.000000
5	25.000000
6	36.000000
7	49.000000
8	64.000000
9	81.000000
10	100.000000

Enter the expression to be evaluated, using *n* for the number of customers. For syntax details and a list of supported functions click on "Help"

n^2

Figure 2.11: Load Dependent editing window


JMVA - PFQN exact solver
File Action Help
Classes Stations Service Times Visits Comment
< Back Next > Solve Exit Help

Visits
Average number of accesses of each class to the station.

*	ClosedClass	OpenClass	Class3
CPU	1.000000	1.000000	1.000000
Disk1	1.000000	1.000000	1.000000
Disk2	1.000000	1.000000	1.000000
Station4	1.000000	1.000000	1.000000

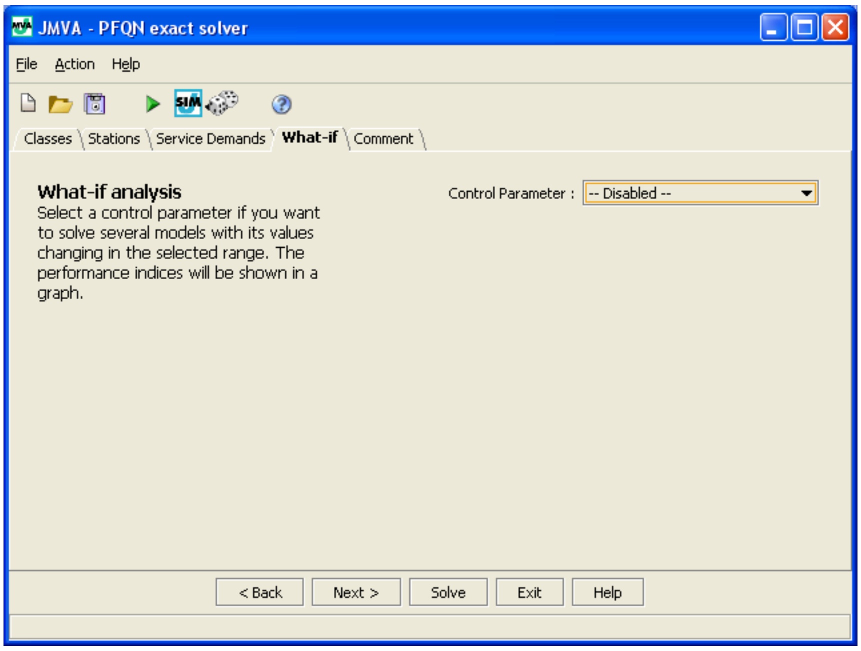
Figure 2.12: Visits Tab

The layout of these tabs is similar to the one of the `Service Demand Tab` described in the previous paragraph. The default value for each element of the Service Times (S) matrix is zero, while it's one for Visits' matrix elements.

In current model contains load dependent stations, the behavior of `Service Times Tab` for their parametrization will be identical to the one described on the previous paragraph for `Service Demands Tab`. On the other hand `Visits Tab` behavior won't change as load dependency is a property of service times and not of visits.

2.2.5 What-if Tab

This Tab is used to perform a what-if analysis, i.e. solve multiple models changing the value of a `control parameter`. In Figure 2.13 is shown this panel when what-if analysis is disabled.


JMVA - PFQN exact solver
File Action Help
Classes Stations Service Demands What-if Comment
< Back Next > Solve Exit Help

What-if analysis
Select a control parameter if you want to solve several models with its values changing in the selected range. The performance indices will be shown in a graph.

Control Parameter : -- Disabled --

Figure 2.13: What-if Tab - Disabled analysis

The first parameter to be set is the `control parameter` i.e. the parameter that will be changed to solve different models in a selected range. Five choices are possible:

Disabled : disables what-if analysis, so only a single queueing network model, specified in the previous steps, will be solved. This is the *default* option.

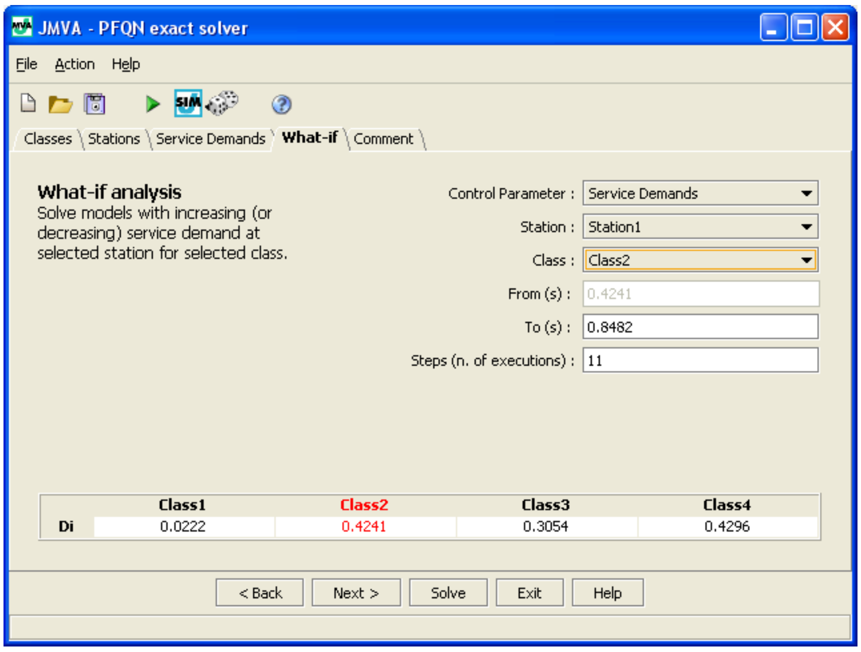
Customer Numbers : different models will be executed by changing the `number of customers` of a single `closed` class or of every closed class proportionally. This option is available only when current model has at least one closed class.

Arrival Rates : different models will be executed by changing `arrival rate` of a single `open` class or of every open class proportionally. This option is available only when current model has at least one open class.

Population Mix : the total number of customers will be kept constant, but the population mix (i.e. the ratio between `number of customers` of selected closed class i and the total number of customers in the system $\beta_i = N_i / \sum_k N_k$). This option is available only when current model has two closed classes.

Service Demands : different models will be solved changing the `service demand` value of a given station for a given class or for all classes proportionally. This option is available only for `load independent` and `delay` stations.

Whenever a control parameter is selected, the window layout will be changed to allow the selection of a valid range of values for it. For example in Figure 2.14 `Service Demands` control parameter was selected. On the bottom of the window, a riepilogative table is presented: depending on selected control parameter, that table is used to show the *initial state* of involved parameters. Every class currently selected for what-if analysis is shown in red.


JMVA - PFQN exact solver
File Action Help
Classes Stations Service Demands What-if Comment
< Back Next > Solve Exit Help

What-if analysis
Solve models with increasing (or decreasing) service demand at selected station for selected class.

Control Parameter : Service Demands
Station : Station1
Class : Class2
From (s) : 0.4241
To (s) : 0.8482
Steps (n. of executions) : 11

	Class1	Class2	Class3	Class4
Di	0.0222	0.4241	0.3054	0.4296

Figure 2.14: What-if Tab - Service Demands

A brief description of each field is now presented:

Station : available only with `Service Demands` control parameter. This combo box allows to select at which station service demand values will be modified.

Class : allow to select for which class the selected parameter will be changed. A special value, namely `All classes proportionally`, is used to modify the control parameter for each class keeping constant the proportion between different classes[2]. This special value is not available in `Population Mix` analysis as we are changing the proportion of jobs between two closed classes.

From : the initial value of what-if analysis. It was chosen to leave this value fixed to the initial value specified by the user in the previous steps to avoid confusions, so this field acts as a reminder. The only exception is when `Population Mix` is changed, in that case it's allowed to modify this value too.

To : the final value of what-if analysis. Please notice that this value can be greater or smaller than `From` value and is expressed in the same measure unit. Whenever `All classes proportionally` option is selected, both `From` and `To` values are expressed as percentages of initial values (specified in the previous steps

[2]for example, in a model with two closed classes with population vector (2,6), the following models can be executed: (1,3), (2,6), (3,9), (4, 12), . . .

and reminded in the table at the bottom of the panel, see Figure 2.14), in the other situations they are considered as absolute values for the chosen parameter.

Steps : this is chosen number of executions i.e. the number of different models that will be solved. When control parameter is `Customer Numbers` or `Population Mix`, the model can be correctly specified only for integer values of population. JMVA will perform a fast computation to find the maximum allowed number of executions given current `From` and `To` values: if user specify a value bigger than that, JMVA will use the computed value.

2.2.6 Comment Tab

In this Tab, a short - optional - comment about the model can be inserted; it will be saved with the other model parameters.

2.2.7 Expression Evaluator

An expression evaluator is used for the definition of service demands or service times of a load dependent station. It allows to specify times as an analytic function of n where n is the number of customer inside the station.

Expression are evaluated using *JEPLite*[3] (Java Math Expression Parser enlited) package which supports all operators enumerated in Table 2.1 and all functions enumerated in Table 2.2.

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

Table 2.1: List of all supported operators ordered by priority

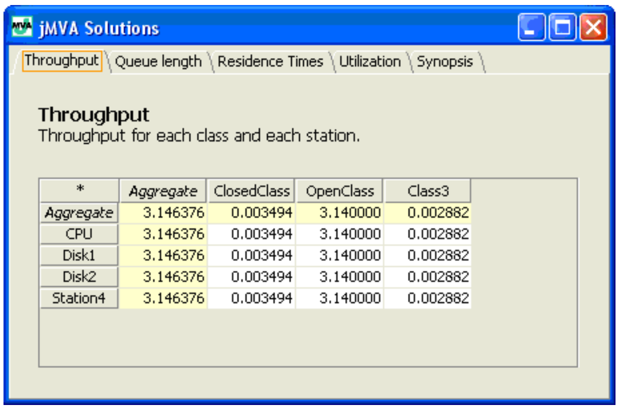
Function	Symbol
Sine	sin()
Cosine	cos()
Tangent	tan()
Arc Sine	asin()
Arc Cosine	acos()
Arc Tangent	atan()
Hyperbolic Sine	sinh()
Hyperbolic Cosine	cosh()
Hyperbolic Tangent	tanh()
Inverse Hyperbolic Sine	asinh()
Inverse Hyperbolic Cosine	acosh()
Inverse Hyperbolic Tangent	atanh()
Natural Logarithm	ln()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

Table 2.2: List of supported functions for the load dependent service times

2.2.8 Model Solution

Use `Solve` command to solve the model. If the model specify a what-if analysis, please refer to subsection 2.2.9. Model results will be shown on a separate window, like the one of Figure 2.15.

[3]http://jeplite.sourceforge.net/



Throughput
Throughput for each class and each station.

*	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

Figure 2.15: Model Solution (Throughput Tab)

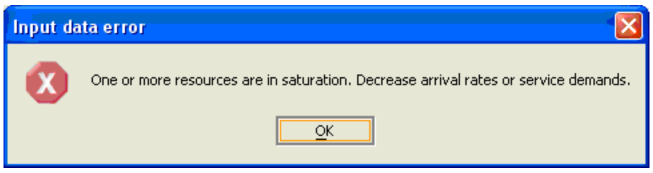
Using the tab selector, all the other computed performance indices can be seen: Throughput, Queue lengths, Residence Times, Utilizations and a synopsis panel with schematic report of input model. Both results and synopsis tab data can be copied to clipboard with `CTRL+C` keyboard shortcut.

When open classes are used, the resource saturation control is performed. For multiple class models, the following inequality must be satisfied:

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

This inequality ensures that no service center is saturated as a result of the combined loads of all the classes. Let us consider, as example, the model with the classes shown in Figure 2.5 with the D-matrix shown in Figure 2.9 Since $\lambda = 3.14 < 3.33 = 1/0.3 = 1/D_{max}$ the model is not in saturation and the `Solve` command will be executed correctly.

In this example, substituting $D_{\text{Disk1-OpenClass}}$ with values $\geq 1/3.14 \approx 0.318$ will cause the saturation of resource *Disk1* and the error message of Figure 2.16 will appear.



One or more resources are in saturation. Decrease arrival rates or service demands.

Figure 2.16: Input data error message

2.2.9 Model Solution - What-if analysis

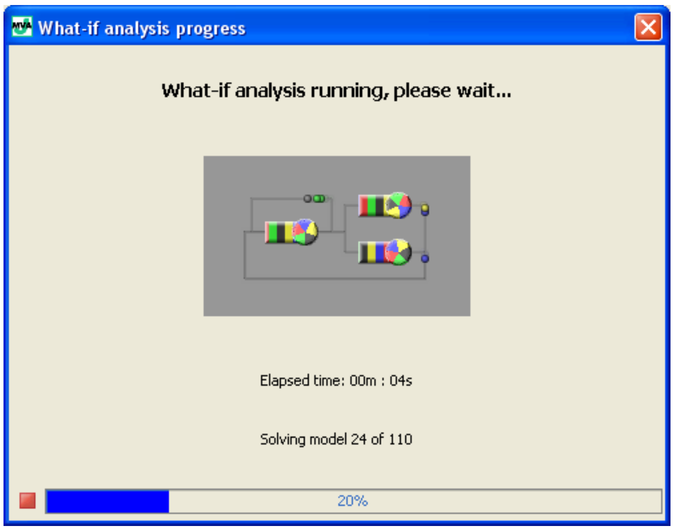
Use `Solve` command to solve the model. During model solution, a progress window, see Figure 2.17, shows up. It displays the cumulative number of models currently solved, the total number of models to be solved and the elapsed time.

At the end of the solution, results will be shown in a separate window, see Figure 2.18. This tab allows to show in a plot the relation between the chosen control parameter (see subsection 2.2.5) and the performance indices computed by the analytic engine.

The combo box `Performance Index` allows to select the performance index to be plotted in the graph, while in the table below, users can select the resource and the class considered.

- The first column is fixed and lists all available colors to be used in the graph.
- The second column, named `Class`, is used to select the class considered in the graph. The special value `Aggregate` is used for the aggregate measure for all classes. If input model is single-class, the class is selected by default for each row.
- The final column, named `Station`, is used to select the station considered in the graph. The special value `Aggregate` is used for the aggregate measure for the entire network. Note that the `Aggregate` value is not valid when the `Utilization` performance index is selected.

In addition to the *center* performance indices (i.e. `Throughput`, `Queue length`, `Residence Times`, `Utilization`), three *system* performance indices are provided in the `Performance Index` combo box (`System Response Time`,



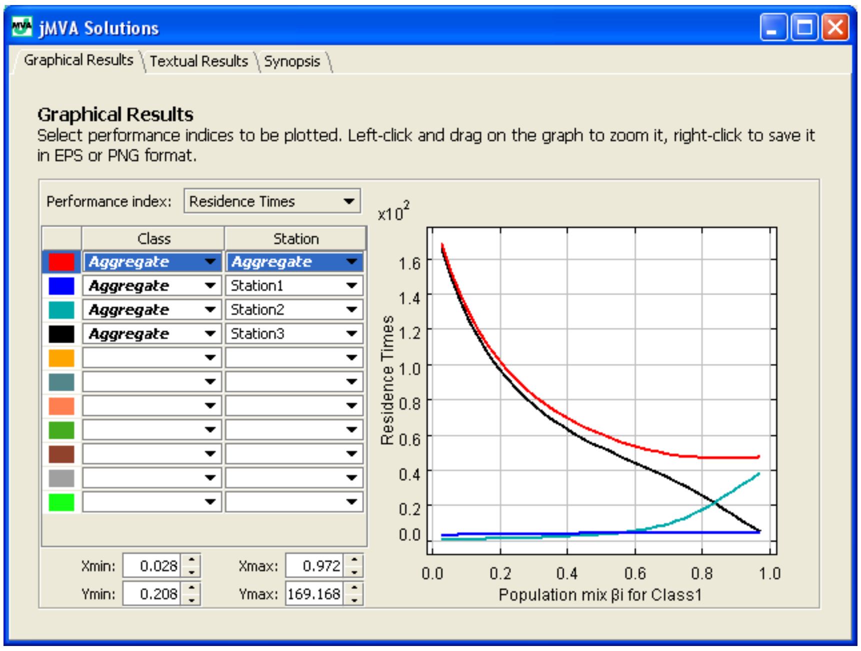
What-if analysis running, please wait...

Elapsed time: 00m : 04s

Solving model 24 of 110

20%

Figure 2.17: Model Solution progress window



Graphical Results

Select performance indices to be plotted. Left-click and drag on the graph to zoom it, right-click to save it in EPS or PNG format.

Performance index: Residence Times

	Class	Station
🟥	Aggregate	Aggregate
🟦	Aggregate	Station1
🟦	Aggregate	Station2
⬛	Aggregate	Station3
🟧		
⬛		
🟧		
🟩		
🟫		
⬜		
🟩		

Xmin: 0.028 Xmax: 0.972

Ymin: 0.208 Ymax: 169.168

Figure 2.18: Model Solution - Graphical Results Tab

`System Throughput`, `Number of Customers`). This *system* indices can be easily obtained by selecting the special `Aggregate` value for both `Class` and `Station` columns of the corresponding center indices (see Appendix A for the definition of the performance indices), but they were provided here as a *shortcut*. As we are referring to aggregate measures, the selection of reference class and station is not significant and, in this case, the table in the left of Figure 2.18 will not be shown.

On the bottom-left corner of the window, users can modify minimum and maximum value of both X and Y axis of the plot. JMVA is designed to automatically best-fit the plot in the window but this controls allow the user to specify a custom range or zoom on the plot. Another fast method to perform a zoom operation is to left-click and drag a rectangle on the graphic window (see Figure 2.19) or right-click on it and select `Zoom in` or `Zoom out` options. To automatically reset the best-fit scale users can right-click on the graphic window and select `Original view` option.

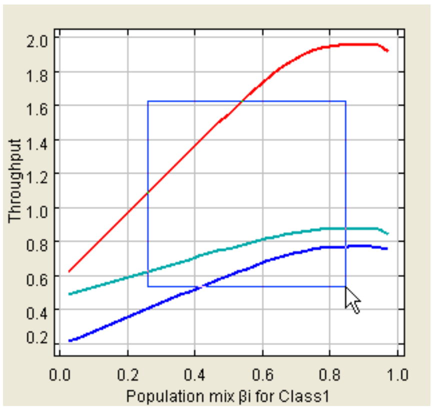
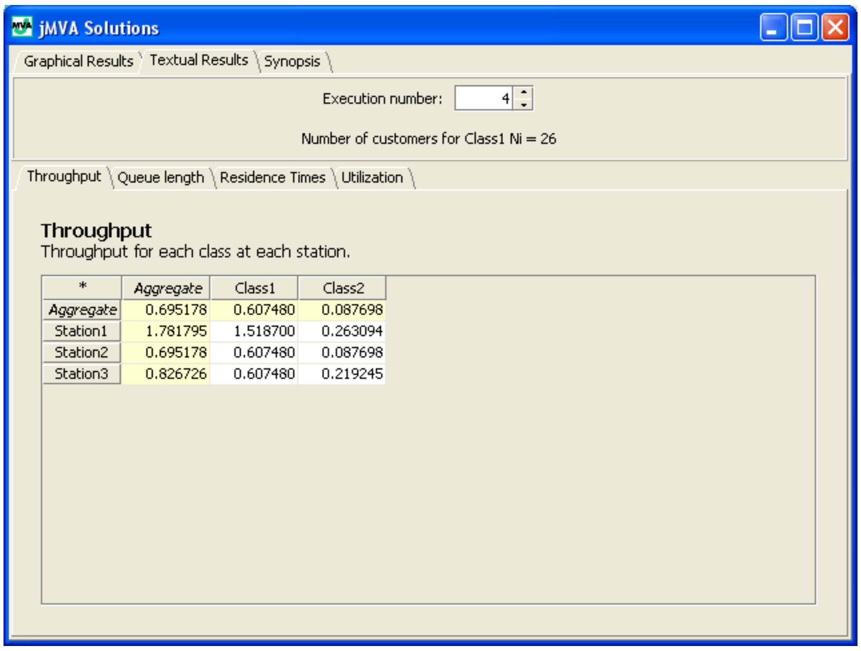


Figure 2.19: Zoom operation on the plot

The graphic window allows to export plots as image to be included in documents and presentations. To save current graph as image, right-click on the graphic and select `Save as...` option. A dialog will be shown to request the name of the file and the format. Currently supported format are Portable Network Graphics - PNG - (raster) and Encapsulated PostScript - EPS - (vectorial, currently only black and white).

The second tab of the solution window, shown in Figure 2.20, is used to display the solutions of each execution of the analytic algorithm.



Execution number: 4

Number of customers for Class1 Ni = 26

Throughput

Throughput for each class at each station.

*	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

Figure 2.20: Model Solution - Textual Results Tab

This Tab has the same structure of the results window without What-if analysis (described in subsection 2.2.8) but allows to select the execution to be shown in the field `Execution Number`. By entering requested execution number in the spinner, or using the *up* and *down* arrows, user can cycle between all the computed performance indices for each execution. Just below the spinner, a label gives information on the value of the control parameter for the currently selected execution.

2.2.10 Modification of a model

To modify system parameters return to the main window and enter new data. After the modifications, if you use `Solve` command, a new window with model result will show. You can `save` this new model with the previous name - overwriting the previous one - or `save` it with a different name or in a different directory.

2.3 Menu entries

2.3.1 File

New

Use this command in order to create a new JMVA model.

Shortcut on Toolbar:	
Accelerator Key:	CTRL+N

Open

Use this command to open an existing model. You can only open one model at time, to edit two or more models start more than one instance of JMVA. If current model was modified since its creation or last save action, a popup window will be shown for confirmation.

It's possible to open not only models saved with JMVA (*.jmva), but also with other programs of the suite (for example JABA *.jaba, JSIM *.jsim and JMODEL *.jmodel). Whenever a foreign data file is opened, a conversion is performed and error/warnings occurred during conversion will be reported in a window.

Models are stored in XML format, see *JMT system manual* for a detailed description.

Shortcut on Toolbar:	
Accelerator Key:	CTRL+O

Save

Use this command in order to save the active document with its current name in the selected directory.

When you save a document for the first time, JMVA displays the Save As dialog box so you can rename your document. If you save a model after its resolution, results are stored with model definition data.

Shortcut on Toolbar:	
Accelerator Key:	CTRL+S

Exit

Use this command in order to end a JMVA session. You can also use the Close command on the application Control menu. If current model was modified since its creation or last save action, a popup window will be shown for confirmation.

Accelerator Key:	CTRL+Q

2.3.2 Action

Solve

Use this command when model description is terminated and you want to start the solution of the model. At the end of the process the window in Figure 2.15 will popup.

Shortcut on Toolbar:	
Accelerator Key:	CTRL+L

Randomize

Use this command in order to insert random values into Service Demands - or Service Times - table. Generated values are automatically adjusted to avoid saturation of resources.

Shortcut on Toolbar:	🎲
Accelerator Key:	CTRL+R

Import in JSIM

This command will import current model into JSIM to solve it using simulator. A simple parallel topology is derived from number of visits at each station and generated model is equivalent to original one.

Shortcut on Toolbar:	SIM
Accelerator Key:	CTRL+G

2.3.3 Help

JMVA Help

Use this command to display application help. From the initial window, you can jump to step-by-step instructions that show how use JMVA and consult various types of reference information.

Once you open Help, you can click the Content button whenever you want to return to initial help window.

Shortcut on Toolbar:	?
Accelerator Key:	CTRL+Q

About

Use it in order to display information about JMVA version and credits.

2.4 Examples

In this section we will describe some examples of model parametrization and solution using MVA exact solver. Step-by-step instructions are provided in five examples:

1. A single class closed model with three load independent stations and a delay service center (subsection 2.4.1)
2. A multiclass open model with two classes and three load independent stations (subsection 2.4.2)
3. A single class closed model with a load dependent station and a delay (subsection 2.4.3)
4. A multiclass mixed model with three stations (subsection 2.4.4)
5. A multiclass closed model where a what-if analysis is used to find optimal Population Mix values (subsection 2.4.5)

2.4.1 Example 1 - A model with a single closed class

Solve the single class model specified in Figure 2.21. The customer class, named *ClosedClass* has a population

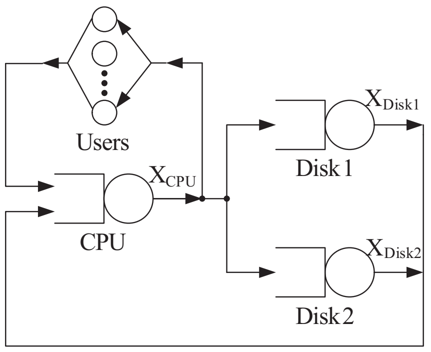


Figure 2.21: Example 1 - network topology

of $N = 3$ customers.

There are four stations, three are of load independent type (named *CPU*, *Disk1* and *Disk2*) and one is of delay type (named *Users*). *Users* delay station represents user's *think time* ($Z = 16$ s) between interaction with the system. Service times and visits for stations are reported in Table 2.3.

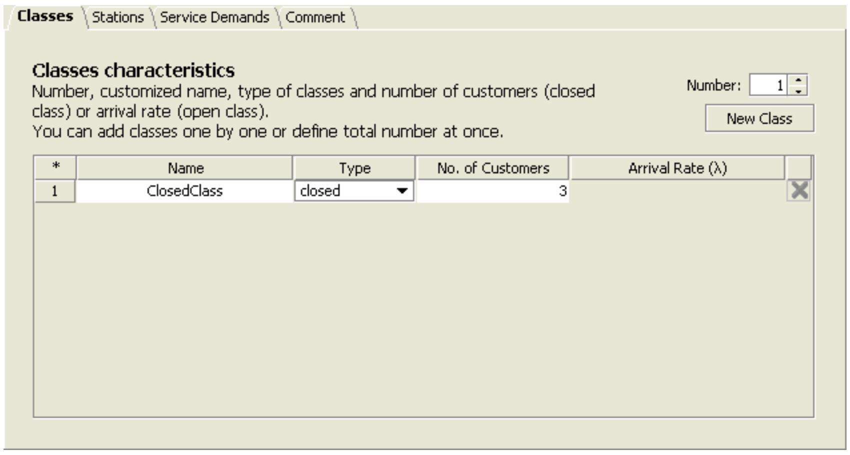
	CPU	Disk1	Disk2	Users
Service Times [s]	0.006	0.038	0.030	16.000
Visits	101.000	60.000	40.000	1.000

Table 2.3: Example 1 - service times and visits

Step 1 - Classes Tab

- use New command to create a new jMVA document
- by default, you have already a `Closed` class
- if you like, substitute default *Class1* name with a customized one (*ClosedClass* in our example)
- complete the table with workload intensity (number of customers). Remember that intensity of a closed class N must be a positive integer number; in this case, 3

At the end of this step, the `Classes Tab` should look like Figure 2.22.



Classes characteristics
Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class).
You can add classes one by one or define total number at once.

*	Name	Type	No. of Customers	Arrival Rate (λ)
1	ClosedClass	closed	3	

Figure 2.22: Example 1 - input data (Classes Tab)

Step 2 - Stations Tab

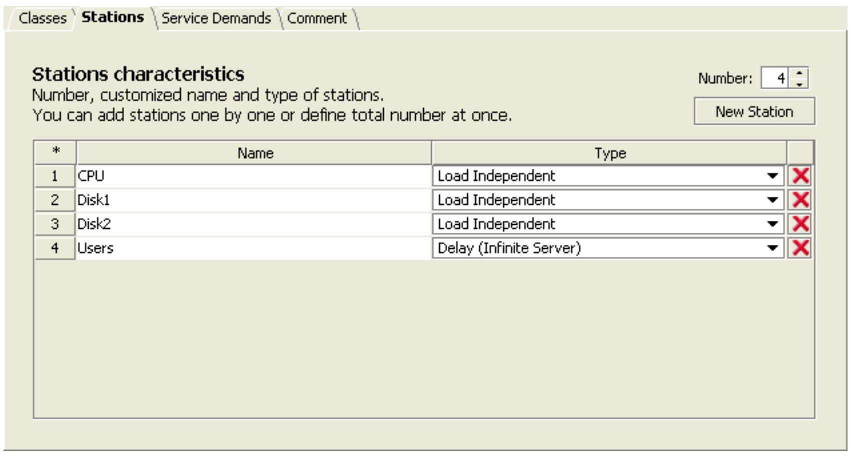
- use `Next` > command to switch to `Stations Tab`
- digit number 4 into stations number textbox or select number 4 using spin controls or push *New Station* button three times. Now your model has four `Load Independent` stations with a default name
- if you want you can change station names. Substitute *CPU* for default name *Station1*, substitute *Disk1* for default name *Station2*, substitute *Disk2* for default name *Station3* and substitute *Users* for default name *Station4*
- change the type of last inserted station; *Users* station is a `Delay (Infinite Server)`

At the end of this step, the `Stations Tab` should look like Figure 2.23.

Step 3 - Service Times and Visits Tabs

- use `Next` > command to switch to `Service Demands Tab`
- press *Service Time and Visit* button as you don't know the Service Demands of the three stations: in this case Service Times and number of Visits should be typed. After button pressure, the `Service Demands Tab` will be hidden and `Service times Tab` and `Visit Tab` will appear
- you can input all Service Times in the table. Remember that Service Time of the *Users* station, of delay type, is *think time Z*, in this case 16 s

At this point, the `Service Times Tab` should look like Figure 2.24.

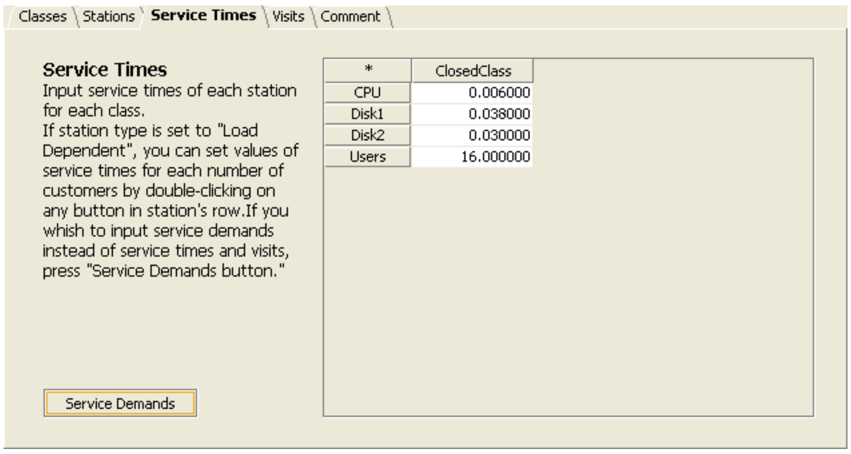


Stations characteristics
Number, customized name and type of stations.
You can add stations one by one or define total number at once.

Number: 4

*	Name	Type
1	CPU	Load Independent
2	Disk1	Load Independent
3	Disk2	Load Independent
4	Users	Delay (Infinite Server)

Figure 2.23: Example 1 - input data (Stations Tab)



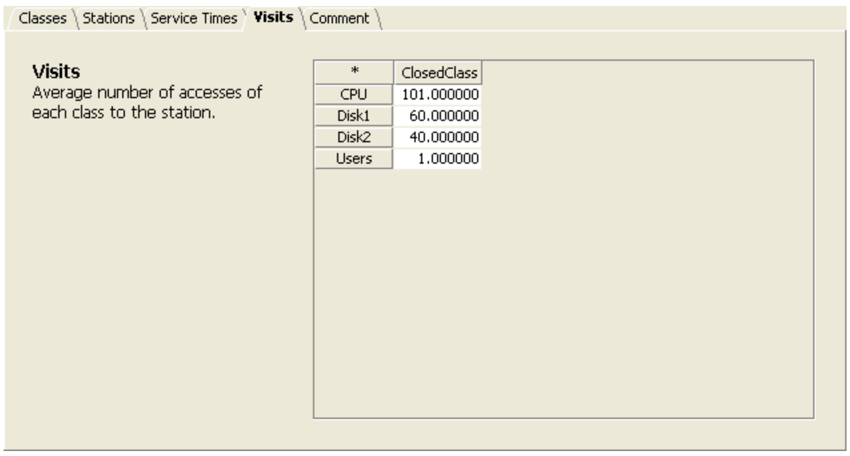
Service Times
Input service times of each station for each class.
If station type is set to "Load Dependent", you can set values of service times for each number of customers by double-clicking on any button in station's row.If you whish to input service demands instead of service times and visits, press "Service Demands button."

*	ClosedClass
CPU	0.006000
Disk1	0.038000
Disk2	0.030000
Users	16.000000

Figure 2.24: Example 1 - input data (Service Times Tab)

- use `Next >` command to switch to `Visits Tab`
- input numbers of visits for all centers in the table. In this case the number of visits of the *Users*, the infinite server station, is equal to 1 since a customer at the end of an interaction with the system visits this station.

At the end of this step, the `Visits Tab` looks like Figure 2.25.



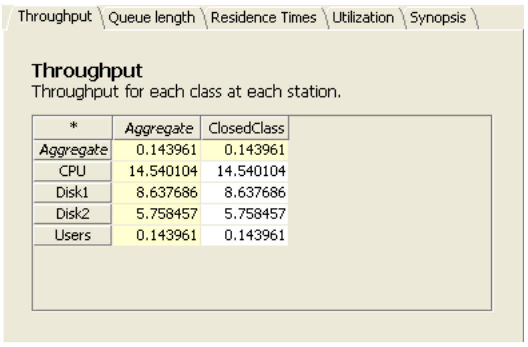
Visits
Average number of accesses of each class to the station.

*	ClosedClass
CPU	101.000000
Disk1	60.000000
Disk2	40.000000
Users	1.000000

Figure 2.25: Example 1 - input data (Visits Tab)

Step 4 - Model Resolution

Use `Solve` command to start the solution of the input model. Model results will be displayed in a new window like the one of Figure 2.26.



Throughput
Throughput for each class at each station.

*	Aggregate	ClosedClass
Aggregate	0.143961	0.143961
CPU	14.540104	14.540104
Disk1	8.637686	8.637686
Disk2	5.758457	5.758457
Users	0.143961	0.143961

Figure 2.26: Example 1 - output data (Throughput Tab)

Since we are considering a single-class model, all results in the column *Aggregate* correspond to the results in the *ClosedClass* column.

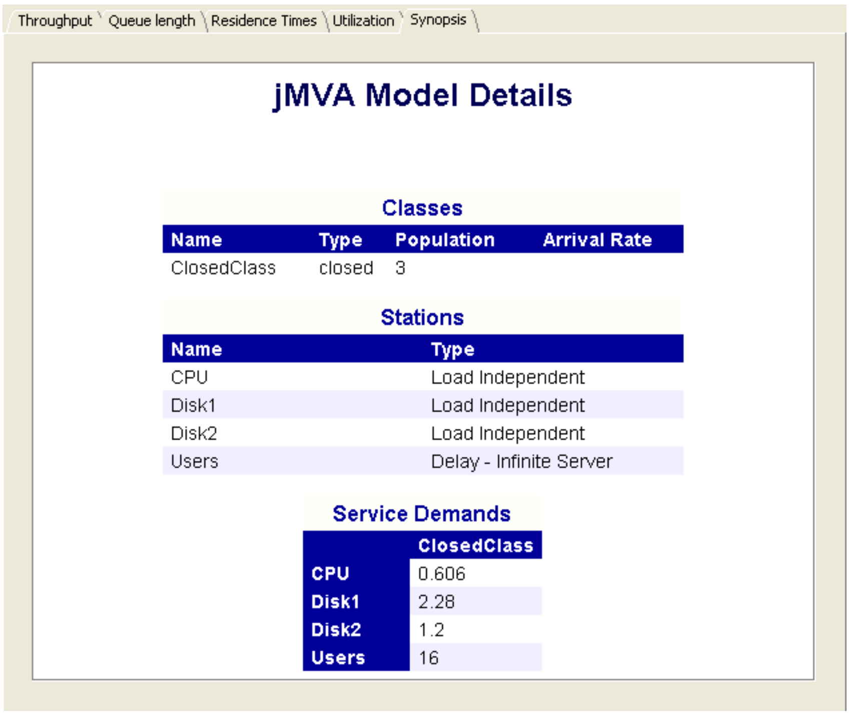
JMVA computes Residence Times W_k, Throughputs X_k, Queue lengths Q_k and Utilizations U_k for all stations. The algorithm begins with the known solution for the network with zero customers, and iterates on N that, in this example, is three. Note that the aggregate *Residence Time* is the *System Response Time* measure and the aggregate *Queue Length* is the average number of customers in the system.

Using tab selector, you can change tab and see Queue length, Residence Times, Utilizations and a synopsis panel with a schematic report of the model (Figure 2.27).

The computed performance indices are shown in Table 2.4.

	Aggregate	CPU	Disk1	Disk2	Users
Throughput [job/s]	0.144	14.540	8.637	5.758	0.144
Queue Length [job]	3.000	0.193	0.410	0.194	2.303
Residence Time [s]	20.839	0.643	2.847	1.349	16.000
Utilization	-	0.087	0.328	0.172	2.303

Table 2.4: Example 1 - model outputs



jMVA Model Details

Classes

Name	Type	Population	Arrival Rate
ClosedClass	closed	3	

Stations

Name	Type
CPU	Load Independent
Disk1	Load Independent
Disk2	Load Independent
Users	Delay - Infinite Server

Service Demands

	ClosedClass
CPU	0.606
Disk1	2.28
Disk2	1.2
Users	16

Figure 2.27: Example 1 - output data (Synopsis Tab)

2.4.2 Example 2 - A model with two open classes

Solve the multiclass open model specified in Figure 2.28. The model is characterized by two open classes A and

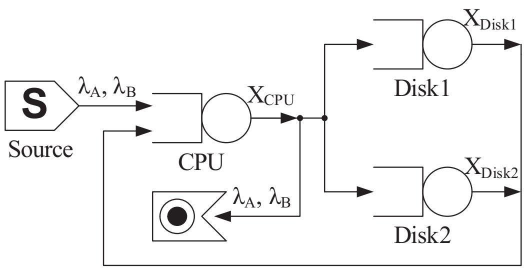


Figure 2.28: Example 2 - network topology

B with arrival rate (the workload intensity λ) respectively of $\lambda_A = 0.15$ job/s and $\lambda_B = 0.32$ job/s. There are three stations of load independent type, identified with names *CPU*, *Disk1* and *Disk2*. Service times and visits for stations are shown in Table 2.5 and Table 2.6.

	CPU	Disk1	Disk2
Class A [s]	0.006	0.038	0.030
Class B [s]	0.014	0.062	0.080

Table 2.5: Example 2 - service times

Since this model is similar to the network of Figure 2.21 solved in subsection 2.4.1, we will show how to easily create it from a saved copy of Example 1:

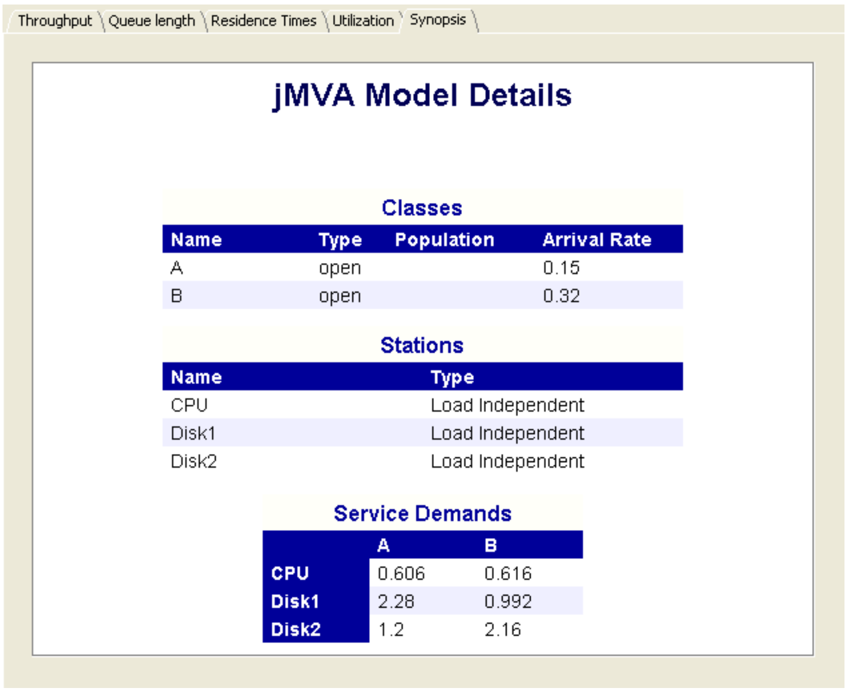
1. Open the saved instance of Example 1 model
2. Go to `Classes Tab`, change *ClosedClass* name to A, change its type to `Open` and set its arrival rate to $\lambda_A = 0.15$ job/s.
3. Click on *New Class* button, sets name of new class to B, change its type to `Open` and set its arrival rate to $\lambda_B = 0.32$ job/s.
4. Go to `Stations Tab` and remove *Users* delay center.
5. Go to `Service Times Tab` and sets service times for Class B according to Table 2.5.
6. Go to `Visits Tab` and sets visits for Class B according to Table 2.6.

	CPU	Disk1	Disk2
Class A	101.0	60.0	40.0
Class B	44.0	16.0	27.0

Table 2.6: Example 2 - number of visits

7. Select `Solve` action.

The `Synopsis Tab` with a schematic report of the model created is shown on Figure 2.29, while the computed performance indices of this model are shown in Table 2.7.



jMVA Model Details

Classes

Name	Type	Population	Arrival Rate
A	open		0.15
B	open		0.32

Stations

Name	Type
CPU	Load Independent
Disk1	Load Independent
Disk2	Load Independent

Service Demands

	A	B
CPU	0.606	0.616
Disk1	2.28	0.992
Disk2	1.2	2.16

Figure 2.29: Example 2 - output data (Synopsis Tab)

	Class A			
	Aggregate	CPU	Disk1	Disk2
Throughput [job/s]	0.150	15.150	9.000	6.000
Queue Length [job]	2.529	0.128	1.004	1.398
Residence Time [s]	16.863	0.851	6.695	9.317
Utilization	-	0.091	0.342	0.180

	Class B			
	Aggregate	CPU	Disk1	Disk2
Throughput [job/s]	0.320	14.080	5.120	8.640
Queue Length [job]	6.575	0.277	0.932	5.366
Residence Time [s]	20.548	0.865	2.913	16.770
Utilization	-	0.197	0.317	0.691

Table 2.7: Example 2 - model outputs

2.4.3 Example 3 - A model with a load dependent station

The network is shown in Figure 2.30. It comprises only two stations: one is of delay type (named *Users*) and the other is a load dependent station (named *Station*). This model has one closed class only with $N = 8$ customers. The user's *think time* is $Z = 21$ s, while the service demands for the load dependent *Station*, shown in Table 2.8, are function of n: number of customers in the station ($D(n) = n + 1/n$).

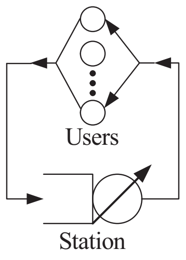


Figure 2.30: Example 3 - network topology

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

Table 2.8: Example 3 - service demands for *Station*, a load-dependent service center

Step 1 - Classes Tab

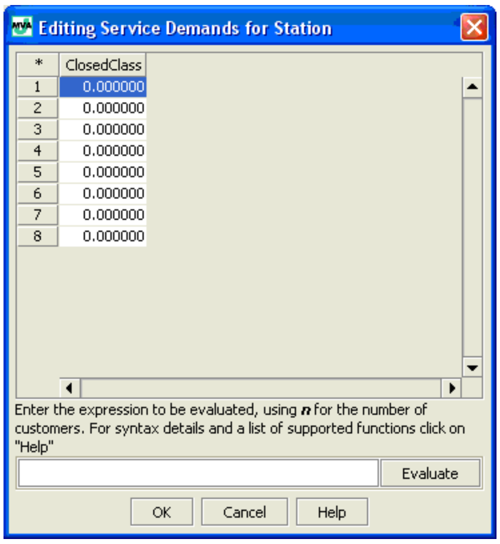
The instructions that are the same given in Step 1 of subsection 2.4.1; in this case N must be 8.

Step 2 - Stations Tab

The instructions are given in Step 2 of subsection 2.4.1; in this case the model has two stations: a Load Dependent station and a Delay Center.

Step 3 - Service Demands Tab

1. use `Next` > command to switch to `Service Demands Tab`
2. double-click on cell with text *LD Setting...* to open the editor of load dependent service demands in a separate window, shown in Figure 2.31.
3. it is not mandatory to insert all values one-by-one. You can click or drag to select cells, enter the expression $n + 1/n$ into the textbox at the bottom of the window and click the *Evaluate* button.
4. at the end of this phase, editor window looks like Figure 2.32. Now you may press *OK* button to confirm changes and return to JMVA main window.



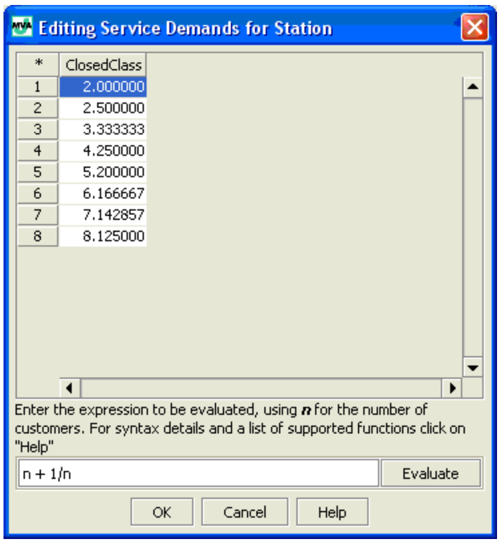
*	ClosedClass
1	0.000000
2	0.000000
3	0.000000
4	0.000000
5	0.000000
6	0.000000
7	0.000000
8	0.000000

Enter the expression to be evaluated, using *n* for the number of customers. For syntax details and a list of supported functions click on "Help"

Figure 2.31: Example 3 - editor for the description of service demands for a load dependent station corresponding to the different number of customers before the parametrization

Step 4 - Model Resolution

Use `Solve` command to resolve the model, results are shown in Table 2.9.



*	ClosedClass
1	2.000000
2	2.500000
3	3.333333
4	4.250000
5	5.200000
6	6.166667
7	7.142857
8	8.125000

Enter the expression to be evaluated, using *n* for the number of customers. For syntax details and a list of supported functions click on "Help"

```
n + 1/n
```

Figure 2.32: Example 3 - editor for the description of service demands for a load dependent station. In this case an arithmetic function has been defined: $S(n) = n + 1/n$

	Aggregate	Station	Users
Throughput [job/s]	0.234	0.234	0.234
Queue Length [job]	8.000	3.080	4.920
Residence Time [s]	34.149	13.149	21.000
Utilization	-	0.810	0.973

Table 2.9: Example 4 - model outputs

2.4.4 Example 4 - A model with one open and one closed class

The mixed queueing network model is shown in Figure 2.33. Workload intensities: the open class has an arrival

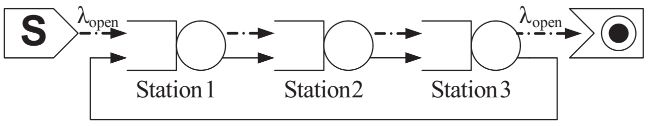


Figure 2.33: Example 4 - network topology

rate $\lambda = 1$ job/s, the closed class has a customers number $N = 57$. Service demands are shown in Table 2.10.

Step 1 - Classes Tab

Follow the instructions of Step 1 in the previous examples; the `Classes Tab` is shown in Figure 2.34.

Step 2 - Stations Tab

Follow the instructions of Step 2 in the previous examples; in this case the model has three Load Independent stations (see Figure 2.35).

Step 3 - Service Demands Tab

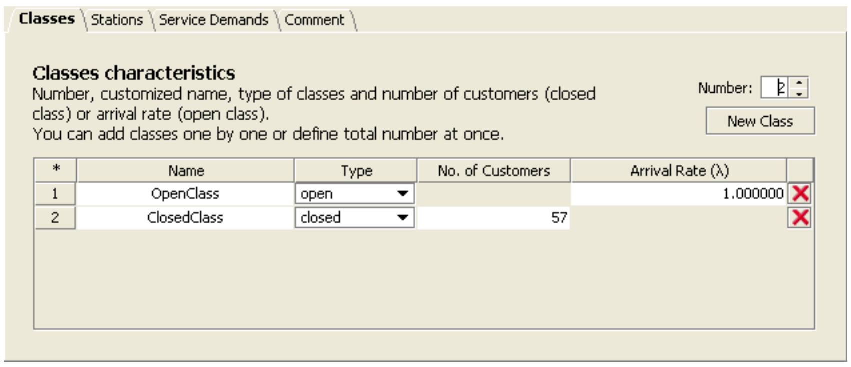
Follow the instructions of Step 3 in the previous examples and define service demands for both classes as illustrated in Table 2.10 (see Figure 2.36).

Step 4 - Model Solution

Use `Solve` command. Results can be verified by computing the *equivalent model*, where the open class "slows down" the closed class by subtracting utilization to it:

	Station1	Station2	Station3
OpenClass [s]	0.5	0.8	0.6
ClosedClass [s]	10.0	4.0	8.0

Table 2.10: Example 4 - service demands

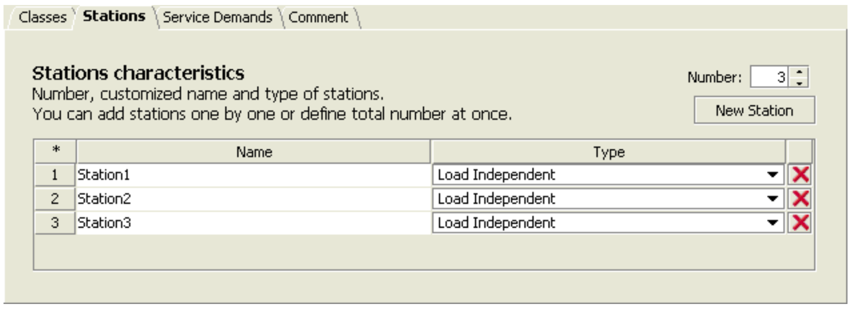


Classes characteristics
Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class).
You can add classes one by one or define total number at once.

Number: 2

*	Name	Type	No. of Customers	Arrival Rate (λ)
1	OpenClass	open		1.000000
2	ClosedClass	closed	57	

Figure 2.34: Example 4 - Class Tab

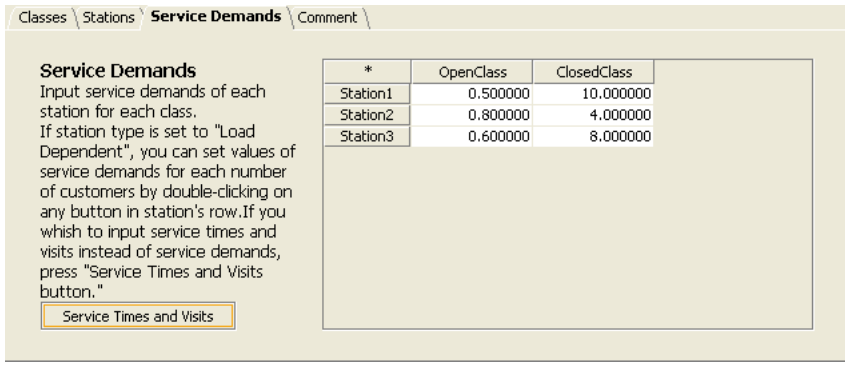


Stations characteristics
Number, customized name and type of stations.
You can add stations one by one or define total number at once.

Number: 3

*	Name	Type
1	Station1	Load Independent
2	Station2	Load Independent
3	Station3	Load Independent

Figure 2.35: Example 4 - Stations Tab



Service Demands
Input service demands of each station for each class.
If station type is set to "Load Dependent", you can set values of service demands for each number of customers by double-clicking on any button in station's row.If you whish to input service times and visits instead of service demands, press "Service Times and Visits button."

*	OpenClass	ClosedClass
Station1	0.500000	10.000000
Station2	0.800000	4.000000
Station3	0.600000	8.000000

Figure 2.36: Example 4 - Service Demands Tab

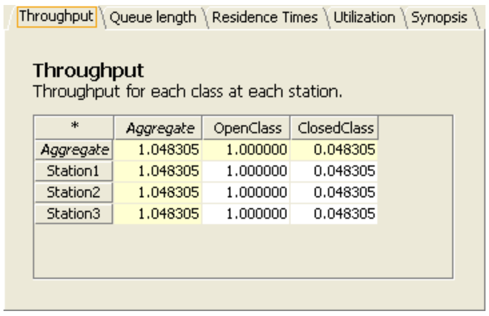
$$
\begin{aligned}
D_1^{eq} &= \frac{D_{1,ClosedClass}}{1-\lambda * D_{1,OpenClass}} = 20s \\
D_2^{eq} &= \frac{D_{2,ClosedClass}}{1-\lambda * D_{2,OpenClass}} = 20s \\
D_3^{eq} &= \frac{D_{3,ClosedClass}}{1-\lambda * D_{3,OpenClass}} = 20s
\end{aligned}
$$

MVA algorithm is used to solve the equivalent closed model. The number of customers of the closed class is 57, and the exact MVA technique should require the solution of other 56 models with smaller population. In this particular case, the formula used to compute the throughput can be simplified because the Service Demands are all equals:

$$
\begin{aligned}
X^{eq}(N) &= \frac{N}{\sum_{k=1}^{3} D_k^{eq} + \sum_{k=1}^{3} [D_k^{eq} * Q_k^{eq}(N-1)]} \\
&= \frac{N}{60 + 20 * \sum_{k=1}^{3} Q_k^{eq}(N-1)} \\
&= \frac{N}{60 + 20 * (N-1)}
\end{aligned}
$$

$$X^{eq}(57) = 0.048305 \text{ job/s}$$

So the throughput measure for the closed class is $X_{ClosedClass} = X^{eq} = 0.048305$ job/s while the throughput for the open class coincide with its arrival rate $X_{OpenClass} = \lambda$. As visits were not specified, they have been considered equal to one: that's why throughput is equal at each station for each class (see Figure 2.37), i.e. the solved model consists of 3 stations that are sequentially connected with feedback.



Throughput
Throughput for each class at each station.

*	Aggregate	OpenClass	ClosedClass
Aggregate	1.048305	1.000000	0.048305
Station1	1.048305	1.000000	0.048305
Station2	1.048305	1.000000	0.048305
Station3	1.048305	1.000000	0.048305

Figure 2.37: Example 4 - throughput

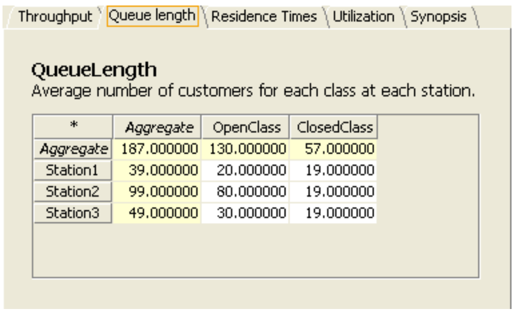
Queue lenghts can be computed with the following formulas:

$$
\begin{aligned}
Q_{k,ClosedClass}(N) &= Q_k^{eq}(N) \\
Q_{k,OpenClass}(N) &= \frac{\lambda * D_{k,OpenClass} * [1 + Q_{k,ClosedClass}(N)]}{1-\lambda * D_{k,OpenClass}}
\end{aligned}
$$

And the results will be equal to the ones shown in Figure 2.38.

2.4.5 Example 5 - Find optimal Population Mix values

Perform a what-if analysis to find the value of population mix that will maximize `System Throughput` and minimize `System Response Time` in the model shown in Figure 2.39. This model has two closed classes (named *Class1* and *Class2*) with a total population of $N = 20$ and three load independent stations (named *Station1*, *Station2* and *Station3*) with the service demands shown in Table 2.11.



QueueLength
Average number of customers for each class at each station.

*	Aggregate	OpenClass	ClosedClass
Aggregate	187.000000	130.000000	57.000000
Station1	39.000000	20.000000	19.000000
Station2	99.000000	80.000000	19.000000
Station3	49.000000	30.000000	19.000000

Figure 2.38: Example 4 - queue lengths

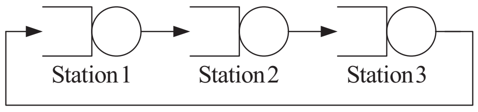


Figure 2.39: Example 5 - network topology

Step 1 - Classes Tab

Follow the instructions of Step 1 in the previous examples; as we will change population mix, initial allocation of the $N = 20$ jobs is irrelevant. For example we can allocate $N_1 = 10$ jobs to *Class1* and $N_2 = 10$ jobs to *Class2*. The `Classes Tab` is shown in Figure 2.40.

Step 2 - Stations Tab

Follow the instructions of Step 2 in the previous examples; in this case the model has three Load Independent stations (see Figure 2.41).

Step 3 - Service Demands Tab

Follow the instructions of Step 3 in the previous examples and define service demands for both classes as illustrated in Table 2.11 (see Figure 2.42).

Step 4 - What-if Tab

1. use `Next` > command to switch to `What-if Tab`
2. Select `Population Mix` as a *control parameter* of the analysis in the combo box, several fields will be shown below.
3. *Class1* is already selected, by default, as reference class for the what-if analysis. This means that β_i values in `From` and `To` fields are referred to *Class1*.
4. By default, JMVA suggests the minimum allowed value of β_1 in the `From` field and its the maximum value in the `To` field[4]. Since we want to find the optimal value in the entire interval, we leave this unchanged.
5. we want to perform the maximum number of allowed executions, so we enter a big number in the `Steps` field (100 for example). JMVA will automatically calculate the maximum number of allowed executions provided that number of customers for each class must be an integer and will report 19.

At the end of this phase, the What-if Tab will look like Figure 2.43.

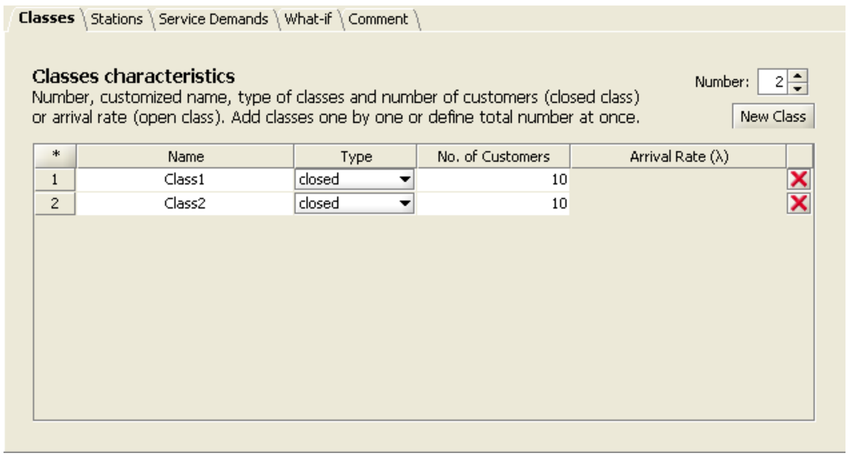
Step 5 - Model Solution

Use `Solve` command. In the `Graphical Results Tab` select `System Response Time` and `System Throughput` as *Performance Index* (Figure 2.44 and Figure 2.45). Zooming on the plot, allows to identify the maximum

[4]Since it is requested that one class has at least one job and customer number must be integer, the minimum value is $1/N$ and the maximum value is $(N-1)/N$.

	Station1	Station2	Station3
Class1 [s]	1.0	5.0	1.0
Class2 [s]	5.0	1.0	5.0

Table 2.11: Example 5 - service demands



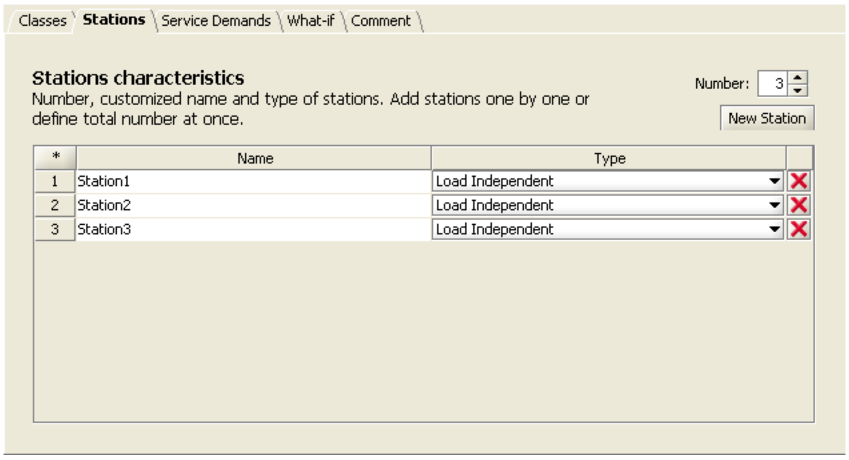
Classes characteristics

Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class). Add classes one by one or define total number at once.

Number: 2

*	Name	Type	No. of Customers	Arrival Rate (λ)
1	Class1	closed	10	
2	Class2	closed	10	

Figure 2.40: Example 5 - Class Tab



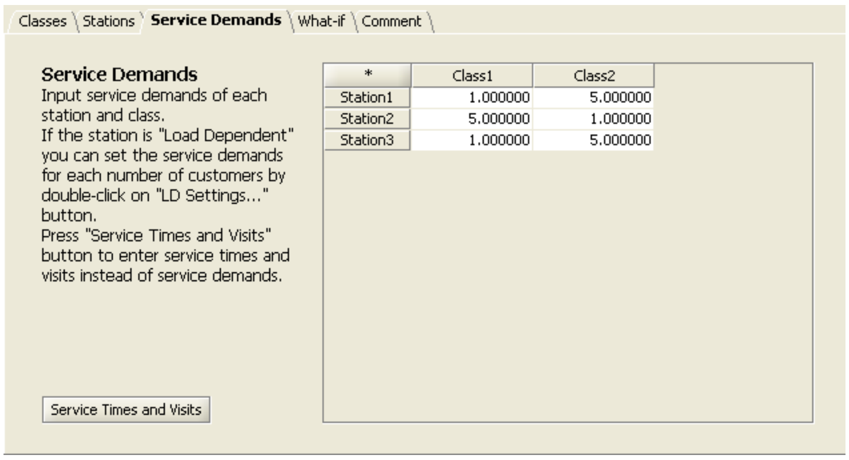
Stations characteristics

Number, customized name and type of stations. Add stations one by one or define total number at once.

Number: 3

*	Name	Type
1	Station1	Load Independent
2	Station2	Load Independent
3	Station3	Load Independent

Figure 2.41: Example 5 - Stations Tab

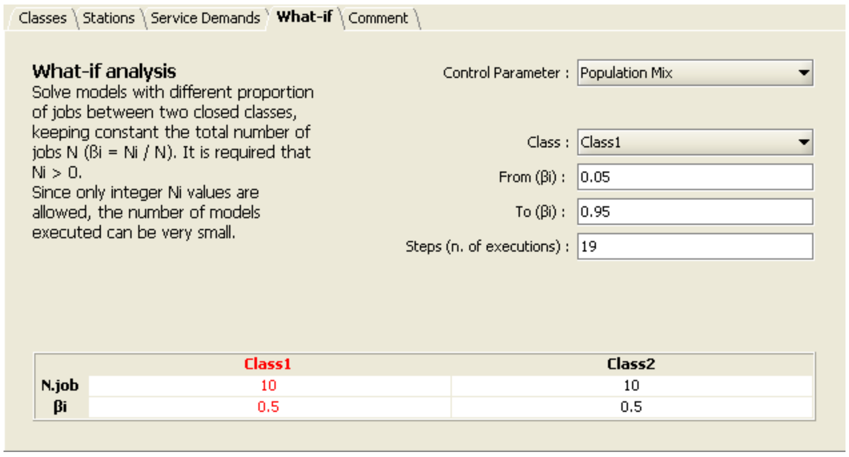


Service Demands

Input service demands of each station and class.
If the station is "Load Dependent" you can set the service demands for each number of customers by double-click on "LD Settings..." button.
Press "Service Times and Visits" button to enter service times and visits instead of service demands.

*	Class1	Class2
Station1	1.000000	5.000000
Station2	5.000000	1.000000
Station3	1.000000	5.000000

Figure 2.42: Example 5 - Service Demands Tab



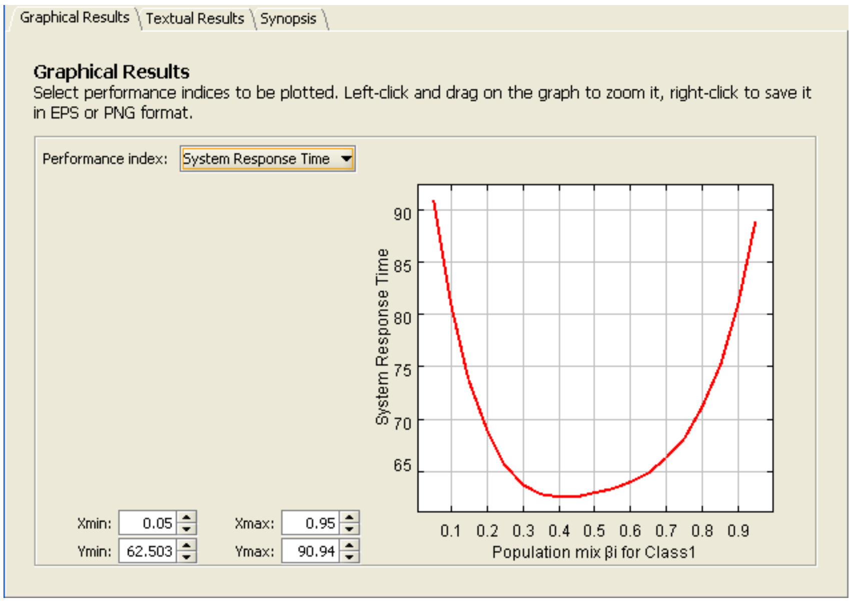
What-if analysis
Solve models with different proportion of jobs between two closed classes, keeping constant the total number of jobs N (βi = Ni / N). It is required that Ni > 0.
Since only integer Ni values are allowed, the number of models executed can be very small.

Control Parameter : Population Mix
Class : Class1
From (βi) : 0.05
To (βi) : 0.95
Steps (n. of executions) : 19

	Class1	Class2
N.job	10	10
βi	0.5	0.5

Figure 2.43: Example 5 - What-if Tab

value of System Throughput (0.32 job/s) and the minimum value of System Response Time (62.50 s). They both corresponds to the execution with population mix $\beta_1 = 0.40$ for *Class1*, which means that the optimal population values are $N_1 = 8$ and $N_2 = 12$.

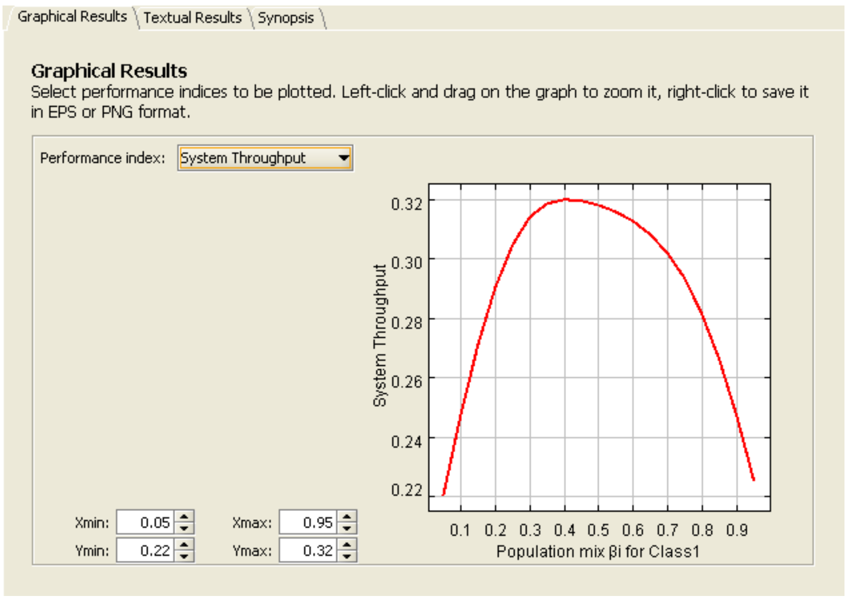


Graphical Results
Select performance indices to be plotted. Left-click and drag on the graph to zoom it, right-click to save it in EPS or PNG format.

Performance index: System Response Time

Xmin: 0.05
Xmax: 0.95
Ymin: 62.503
Ymax: 90.94

Figure 2.44: Example 5 - System Response Time



Graphical Results
Select performance indices to be plotted. Left-click and drag on the graph to zoom it, right-click to save it in EPS or PNG format.

Performance index: System Throughput

Xmin: 0.05
Xmax: 0.95
Ymin: 0.22
Ymax: 0.32

Figure 2.45: Example 5 - System Throughput

Appendix A

Basic Definitions

Customer : or *job* or *transaction* is the element that will require service from our network. For example, it can be an *http request*, a *database query*, or an *ftp download command.*

Class : a group of elements that are statistically equal. The customers of a class have the same workload intensity and the same average service demands.

Station : or service center, it represents a system resource. Customers arrive at the station and then, if necessary, wait in queue, receive service from server then depart.

Workload Intensity (λ or N) : rate at which customers of a given class arrive at the system. Each class has its own workload intensity. If a class is closed the constant *number of customers* N that are present in the system must be provided, if a class is open the *arrival rate* λ of customers to the system is requested.

Population Mix (β_i) : the ratio between *number of customers* of closed class i and the total number of customers in the system: $\beta_i = N_i / \sum_k N_k$

Service Time (S_k): average time of service required at each visit to resource k; it is computed as the ratio of the busy time to the number of system completions. It is an alternatively way to describe the service requirement.

Visit (number of -) (V_k) : average number of visits that a customer makes at station k during a complete execution; it is computed as the ratio of the number of completions at resource k to the number of system completions. If resource k is a delay center representing a client station, it is a convention assign a unitary value to number of visits to this station.

Service Demand (D_k) : average service requirement of a customer, that is the total amount of service required by a complete execution at resource k. In the model it is necessary to provide separate service demand for each pair service center-class. It is given by $D_k = V_k * S_k$.

Throughput (X_k) : rate at which customers are executed by station k, that is the number of completions in a time unit.

Queue length (Q_k) : average number of customers at station k, either waiting in queue and receiving service.

Response Time (R_k): average time interval between the instant a customer arrives at station k and the instants it terminate its servicing.

Residence Time (W_k) : average time that a customer spent at station k during a complete interaction with the system. It includes time spent queueing and time spent receiving service. It does not correspond to *Response Time* R_k of a station since $W_k = R_k * V_k$.

Utilization (U_k) : proportion of time in which the station k is busy or, in the case of a delay center, is the average number of customers in the station (see Little Law [Lit61]).

System Throughput (X) : rate at which customers perform an entire interaction with the system. This is the aggregate measure of *Throughput*: $X = X_k / V_k$.

System Response Time (R): correspond to the intuitive notion of response time perceived by users, that is, the time interval between the instant of the submission of a request to a system and the instant the corresponding reply arrives completely at the user. It is the aggregate measure of *Residence Times*: $R = \sum_k W_k$.

Average number of customers in the system (N): the average number of customer in the system, either waiting in queue or receiving service. It is the aggregate measure of *Queue Length*: $N = \sum_k Q_k$.

Bibliography

[BBC$^+$81] G. Balbo, S.C. Bruell, L. Cerchio, D. Chialberto, , and L. Molinatti. *Mean value analysis of closed load dependent queueing networks*. Dipartimento di Informatica, Universit di Torino, Oct. 1981.

[BCMP75] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. *Open, Closed and Mixed Networks of Queues with Different Classes of Customers*. J. ACM, April 1975.

[Lit61] John D. C. Little. *A Proof of the Queueing Formula L= λ W. Operations Research*, 9:383–387, 1961.

[LZGS84] E.D. Lazowska, J. Zahorjan, G.S. Graham, and K. Sevcik. *Quantitative System Performance*. Prentice-Hall, 1984.

[RL80] M. Reiser and S.S. Lavenberg. *Mean-Value Analysis of Closed Multichain Queuing Networks*. J. ACM, Vol 27, No 2, pp 313-322, April 1980.