



POLITECNICO DI MILANO

**SCHOOL OF INDUSTRIAL AND INFORMATION
ENGINEERING**

MSc in Computer Science and Engineering

Extending the Graphical User Interface of Java Modeling Tools

Supervisor:

Prof. Marco GRIBAUDO

Master Thesis by:

Emma BORTONE - 10673695

Academic year 2019-2020

Abstract [English]

Computer and communication systems have grown more and more complex, and their place is central in the business world. Understanding those systems is therefore necessary in order to be able to answer the questions that inevitably rise during the lifespan of a system: which resources should be increased, and to what extent, in order to respond to an augmentation of demand? What causes a system to slow down? The need to answer those questions leads to the development of tools and techniques to simulate systems. Those techniques for performance evaluation require the use of analytical or simulative methods and are often based on queuing networks models or on petri nets. It is in this context that the Java Modelling tools project started at Politecnico di Milano in 2002. Java Modelling tools (JMT) is a free open source tool suite for modelling computer and communication systems, it is now developed in cooperation with Imperial college London. Java Modelling Tools contains six tools: JSIMGraph, SIMwiz, JMVA, JABA, JWAT and JMCH. JSIMGraph provides a graphical interface that allows an easy description of the network layout and a nicer visualization of complex systems. JSIMGraph let the user build the network by selecting stations (queues, routers, delays...), connecting them and defining the input parameters. Based on the feedback provided by the users of JSIMGraph, the graphical user interface was extended by introducing a new type of connection with a customizable shape and by allowing the rotation of stations. Those extensions were able to make possible the construction of networks with a nicer and clearer layout, improving the efficiency of creating network with JSIMGraph.

Key Words

Computer system; Network; Modelling; Performance evaluation; Graphical user interface; Java modelling tools; Bezier; User experience; Software; Swing

Abstract [Italiano]

I sistemi informatici e di comunicazione sono diventati sempre più complessi e il loro posto è centrale nel mondo degli affari. La comprensione di questi sistemi è quindi necessaria per poter rispondere alle domande che inevitabilmente sorgono nel corso della vita di un sistema: quali risorse devono essere aumentate, e in che misura, per rispondere ad un aumento della domanda? Cosa provoca il rallentamento di un sistema? La necessità di rispondere a queste domande porta allo sviluppo di strumenti e tecniche per simulare i sistemi. Queste tecniche di valutazione delle prestazioni richiedono l'uso di metodi analitici o simulativi e sono spesso basate su modelli di reti di code o su reti di petri. È in questo contesto che il progetto Java Modelling tools è iniziato al Politecnico di Milano nel 2002. Java Modelling tools (JMT) è una suite di strumenti open source per la modellazione di sistemi di computer e di comunicazione, sviluppata in collaborazione con l'Imperial college di Londra. Java Modelling Tools contiene sei strumenti: JSIMGraph, SIMwiz, JMVA, JABA, JWAT e JMCH. JSIMGraph fornisce un'interfaccia grafica che permette una facile descrizione del layout di rete e una più piacevole visualizzazione di sistemi complessi. JSIMGraph permette all'utente di costruire la rete selezionando le stazioni (code, router, delay...), collegandole e definendo i parametri di ingresso. Sulla base dei feedback forniti dagli utenti di JSIMGraph, l'interfaccia grafica è stata estesa introducendo un nuovo tipo di connessione con una forma personalizzabile e permettendo la rotazione delle stazioni. Tali estensioni sono state in grado di rendere possibile la costruzione di reti con un layout più bello e chiaro, migliorando l'efficienza della creazione di reti con JSIMGraph.

Key Words

Sistema informatico; Rete; Modellazione; Valutazione della performance; Interfaccia grafica; Java modelling tools; Bezier; Esperienza utente; Software; Swing

Table of contents

I.	Introduction.....	1
1.	Purpose.....	1
2.	Definitions, Acronyms, Abbreviations.....	1
3.	Document Structure.....	1
II.	Project Background	2
1.	The need for computer systems modeling tools.....	2
2.	Other existing tools for modelling computer systems.....	3
i.	GreatSPN (GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets)	3
ii.	Möbius.....	4
iii.	PEPA eclipse plug-in	4
iv.	Platform Independent Petri net Editor 2 (PIPE)	5
v.	PRISM	6
vi.	Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE).....	7
vii.	TANGRAM II.....	8
III.	Description of Java Modelling Tools (JMT).....	9
1.	Brief history of JMT	9
2.	Description of JMT	9
i.	JSIMwiz	10
ii.	JMVA.....	10
iii.	JABA.....	11
iv.	JWAT.....	11
v.	JMCH.....	11
vi.	JSIMGraph	12
3.	Study case using JSIMGraph.....	13
i.	First infrastructure: A computing server and a storage server equipped with two disks.....	13
ii.	Second infrastructure: Study of the behavior of a station after a growth of service time ...	16
IV.	Improving JSIMGraph	20
1.	The graphical user interface of JSIMGraph	20
2.	Aspects of JSIMGraph that could be improved	20
i.	The automatic connection shapes generating a confusing layout.....	20
ii.	The input and output overlap	21
iii.	The impossibility of quickly deleting an arc.	22
iv.	The impossibility to rotate stations.....	22
v.	The difficulty to properly and quickly align stations	23

3.	Proposed improvements	23
4.	Requirements	24
V.	Implementation of the extensions	26
1.	Modelling of Bezier curves	26
i.	Definition of Bezier curves	26
ii.	The implementation	27
iii.	New format for the xml file	28
2.	The process of drawing of Bezier connections.....	30
3.	Modifying the connection shapes	32
i.	Moving controls points.....	32
ii.	Adding intermediary points.....	35
iii.	Deleting control points.....	38
iv.	Adding tangents to sharp intermediary points	41
v.	Breaking the tangent symmetry.....	43
vi.	Breaking the connection continuity	46
vii.	Possibility to switch between connection types.	49
4.	Station rotation	49
i.	The image rotation	50
ii.	The ports rotation	50
iii.	The rotation of connections shapes	52
5.	Grid snapping.....	54
VI.	Other extensions ideas.....	54
i.	Add identifier to broken connections	54
ii.	Possibility of deleting a connection and creating a new one by dragging the connection arcs.55	
VII.	Conclusion	56

I. Introduction

1. Purpose

The purpose of this document is to present the extensions added to the graphical user interface of Java Modelling Tool. Java Modelling Tools, abbreviated JMT, is a free open source tool suite for modelling computer and communication systems. This document focuses on the tool JSIMGraph describing its purpose, its functionalities, its issues and how the extensions developed during the thesis work described in this document solves them. The work presented in this document is based on the version 1.0.5 of JMT.

2. Definitions, Acronyms, Abbreviations

Acronym or abbreviation	Definitions
GUI	Graphical user interface
MVA	Mean value analysis
JMT	Java Modelling Tools
GSPN	Generalized Stochastic Petri Net
SWN	Stochastic Well-Formed Net
WMT	Windows Modeling Tool
RECAL	Recursion by Chain Algorithm
MoM	Method of Moments
CoMoM	Class oriented Method of Moments
FCR	FCR finite capacity region

3. Document Structure

The document is organized as follows:

Section 1, Introduction, describes the content of this document.

Section 2, Project background, describes the context that created the need for a computer network modelling tool.

Section 3, Description of Java Modelling tools, provides a detailed description of JMT.

Section 4, Improving JSIMGraph, highlights the features of JSIMGraph that cause issues and suggests extensions to solves them.

Section 5, Implementation of the extensions, describes how the extensions were implemented.

Section 6, Other extensions ideas, suggests other extensions to further improve the tool JSIMGraph.

Section 7, Conclusion, concludes on how the extensions improve the GUI of the tool.

II. Project Background

This section explains why modelling tools such as JMT are needed, and provides examples of the type of questions that can be answered by using modelling tools.

1. The need for computer systems modeling tools

The complexity of computer and communication systems, their rapidity of evolution and their prominent place in the business world leads to the need for tools and techniques to simulate systems and understand their behavior. Those techniques for performance evaluation require the use of analytical or simulative methods. They are often based on queuing networks models or on petri nets.

Understanding those systems is necessary in order to be able to answer the questions that will inevitably rise during the lifespan of a system. The following examples provide an illustration of the type of question that modelling and simulation can solve:

- The manufacturer of a turn-key medical information system needs an efficient way to size systems in preparing bids. Given estimates of the arrival rates of transactions of various types, this vendor must project the response times that the system will provide when running on various hardware configurations.

- A stock exchange intends to begin trading a new class of options. When this occurs, the exchange's total volume of options transactions is expected to increase by a factor of seven. Adequate resources, both computer and personnel, must be in place when the change is implemented.¹

Those questions do not have trivial answers and errors could result in extra costs or even compromise the usability of the system. In order to answer those questions, two different techniques could be used: experimentation and modeling techniques. In order to perform experimentation and measurements, the system needs to already exist; and this is not the case during the design of a new system. Experimentation is also often prohibitively expensive. On the opposite, modelling techniques are cheaper and require only a model of the system. By using a simplified representation of the system, it is then possible to understand and evaluate it. Queuing networks and petri nets are special types of models.

A Petri net is a mathematical model used to describe concurrent and distributed systems. It is a directed bipartite graph in which the arcs connect places (represented by circles) and transitions (represented by bars).

A queuing model represents a computer system as a network of queues. This type of model is composed of service centers, queues and customers that can be parametrized accordingly to the specificities of the real system.

Both Petri nets and queuing networks require the use of a software in order to perform the simulation of the system.

¹ LAZOWSKA E. D., ZAHORIAN J., GRAHAM G. S., SEVCIK K. C. (1984). *Quantitative System Performance - Computer System Analysis Using Queueing Network Models*. Prentice-Hall

2. Other existing tools for modelling computer systems

Many tools are available for the modelling of computer systems, they all have the same global goal as JMT: modelling computer and communication network in order to analyze them and predict their behavior ; but they differ in terms of functionalities and modeling approaches.² In this section, other tools than JMT will be briefly presented. The list of tools presented in this section does not pretend to be exhaustive or a ranking, and it contains only tools that provide a GUI.

i. GreatSPN (GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets)

GreatSPN is a software developed by the *Dipartimento di informatica* at the *Università di Torino*, Turin, Italy in cooperation with the *Dipartimento di Informatica* at the *Università del Piemonte Orientale*, Alexandria, Greece.³

GreatSPN is a tool for solving Generalized Stochastic Petri Nets (GSPNs) and Stochastic Well-Formed Nets (SWNs). It can perform the modeling, validation, and performance evaluation of distributed systems (by means of state-space analysis, simulation, structural analyses and Markov chain generation).⁴

The figure bellow (Figure 1) shows the GUI of the software GreatSPN.

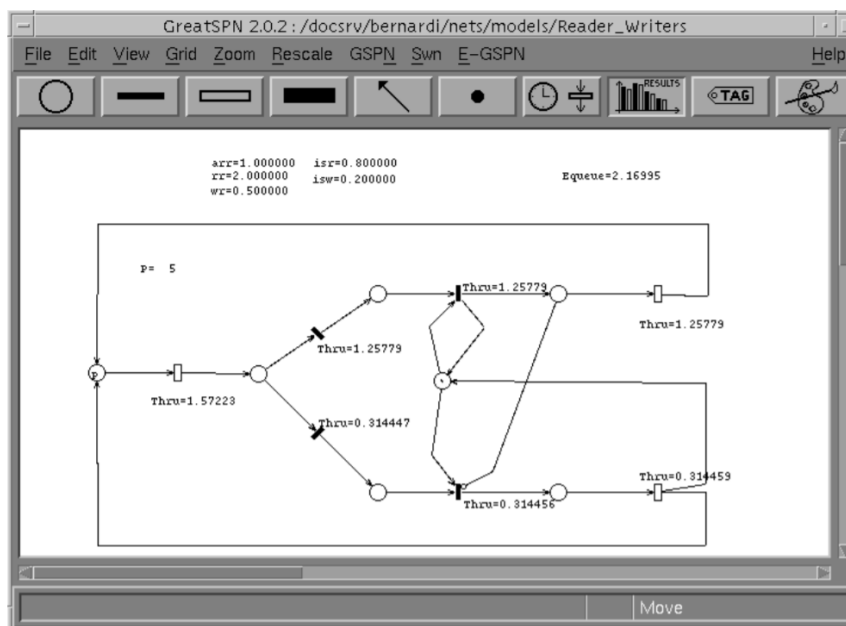


Figure 1: GUI of GreatSPN

Source : <http://www.di.unito.it/~greatspn/index.html#GreatManual>

² CASALE G., MUNTZ R. R., SERAZZI G., (2009), *Special Issue on Tools for Computer Performance Modeling and Reliability Analysis*

³ DIPARTIMENTO DI INFORMATICA Università di Torino, (2008), *GreatSPN*, <http://www.di.unito.it/~greatspn/index.html>

⁴ BABAR J., BECCUTI M., DONATELLI S., MINER A., (2010), *GreatSPN Enhanced with Decision Diagram Data Structures*.

ii. Möbius

Möbius is a software for modeling complex systems developed by the *Perfomability Engineering Research Group (PERFORM)* in the *Center for Reliable and High Performance Computing* at the University of Illinois at Urbana-Champaign

It can be used for studying the reliability, availability, and performance of computer and communication systems but it can also be used for other types of discrete-event systems such as biochemical reactions within genes. This wide utilization range is made possible by the fact that Möbius supports multiple modelling languages and multiple solution techniques. It is possible to choose between graphical or textual representations and between the following model types: stochastic extensions to Petri nets, Markov chains and extensions, and stochastic process algebras.⁵⁶

The figure bellow (Figure 2) shows the GUI of the software Möbius.

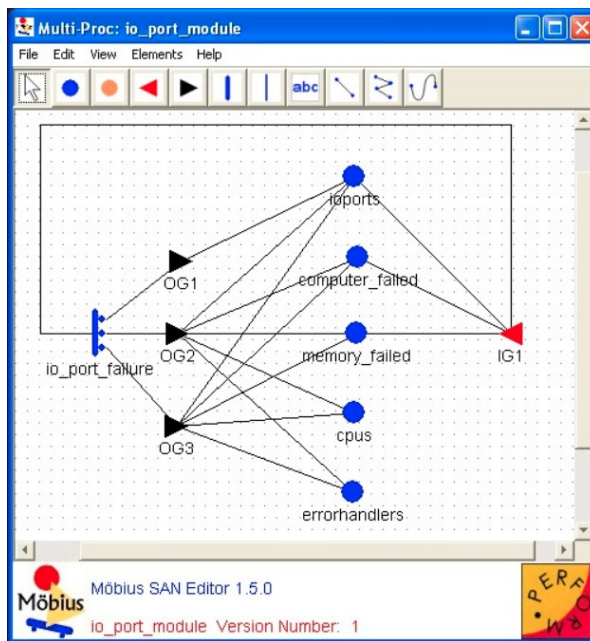


Figure 2: GUI of Möbius

Source : <https://www.mobius.illinois.edu/images/atomic-san-multiproc.jpg>

iii. PEPA eclipse plug-in

The *Performance Evaluation Process Algebra (PEPA)* is a formal language that allows the quantitative analysis of systems. This language was defined by Jane Hilston in 1994 for her PhD thesis at the University of Edinburg.⁷

⁵ COURTNEY T., DALY D., DERISAVI S., GAONKAR S., GRIFFITH M., LAM V., SANDERS H. W., (2004), *The Möbius Modeling Environment: Recent Developments*.

⁶ Möbius, (2020), *Möbius – Model-based environment for validation of system reliability, availability, security, and performance*, <https://www.mobius.illinois.edu/research.php>

⁷ Laboratory for Foundations of Computer Science, The University of Edinburg, (2011), *PEPA – Performance Evaluation Process Algebra*, <http://www.dcs.ed.ac.uk/pepa/>

The PEPA eclipse plug-in allows the Markovian, static and ODE analysis of PEPA models. The tool runs inside the eclipse framework. It provides a user-friendly graphical interface and an API that exposes the PEPA library to third-party Eclipse plug-ins^{8 9}.

The screenshot below (Figure 3) shows a PEPA model with the experimentation wizard. In this example, the graph on the bottoms shows the throughput evolution depending on the variation of a specific input parameters.

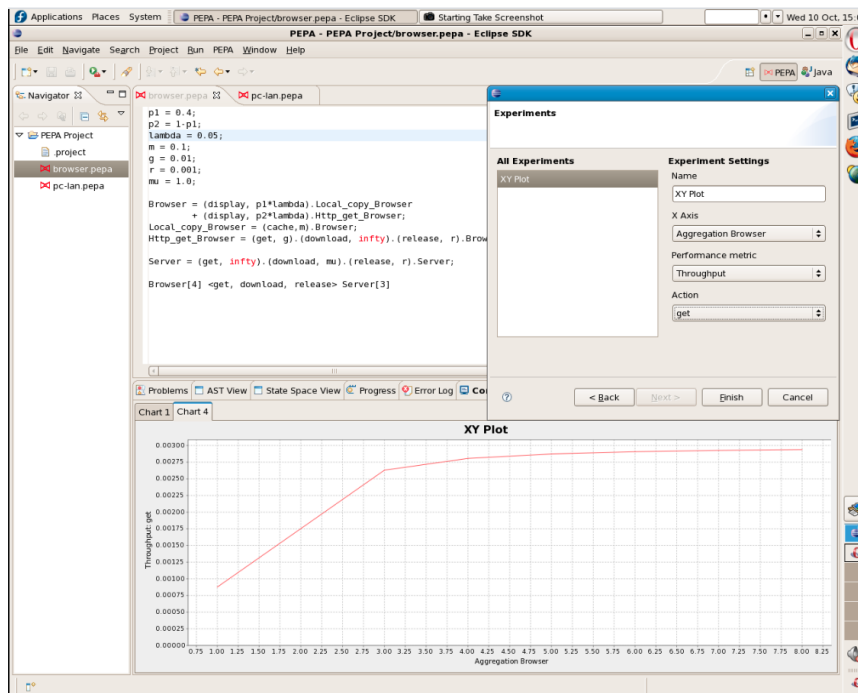


Figure 3: GUI PEPA Eclipse Plug-in

Source : <http://www.dcs.ed.ac.uk/pepa/tools/plugin/screenshots.html>

iv. Platform Independent Petri net Editor 2 (PIPE)

The project PIPE and then PIPE2 was created in 2002 as a MSc. Group Project called *The Platform Independent Petri net Editor PIPE* at the *Department of Computing* at Imperial College London. It is still currently in development at the college¹⁰.

PIPE2 is an open source and platform independent tool for the modelization and evaluation of Petri nets, Model Place transition nets and Generalised Stochastic Petri nets.¹¹

PIPE2 also offers an intuitive graphical user interface (see Figure 4):

⁸ Anonymous, (2009), *The PEPA Plug-in Project*, <http://www.dcs.ed.ac.uk/pepa/tools/plugin/index.html>

⁹ SMITH M J. A., (2010), *A Tool for Abstraction and Model Checking of PEPA Models*

¹⁰ Anonymous, *PIPE2 - Platform Independent Petri net Editor 2* <http://pipe2.sourceforge.net/index.html>

¹¹ BONET P., LLADO C.M., PUIJANER R., KNOTTENBELT W.J., (2007), *PIPE v2.5: A Petri Net Tool for Performance Modelling*.

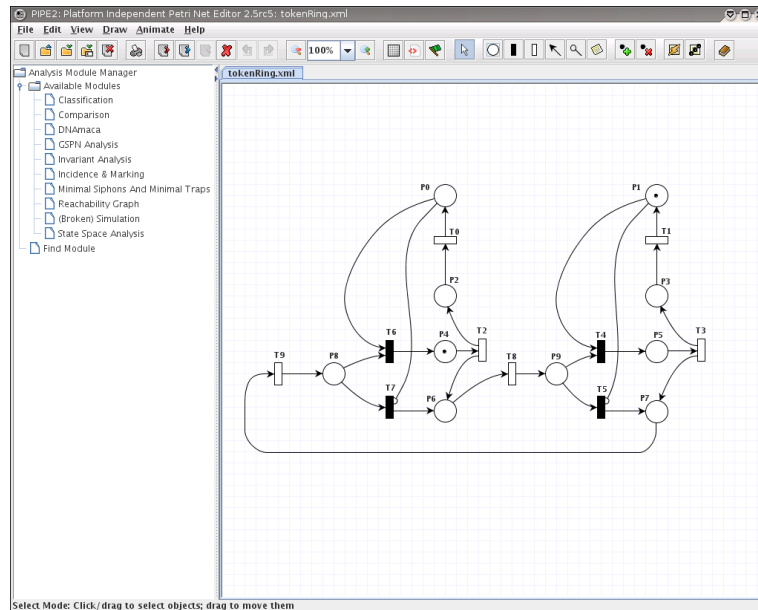


Figure 4: GUI PIPE2

Source : <http://pipe2.sourceforge.net/images/screenshot.png>

v. PRISM

The PRISM open source tool was created and is still actively maintained by Dave Parker (University of Birmingham), Gethin Norman (University of Glasgow) and Marta Kwiatkowska (University of Oxford)¹².

PRISM is a tool for modelling and analyzing systems with random or probabilistic behavior. It can analyze discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), Markov decision processes (MDPs), probabilistic automata (PAs), probabilistic timed automata (PTAs) and the extensions of these models with costs and rewards.¹³ This tool has a wide application domain, from communication, security and multimedia protocols to randomised distributed algorithms and biological systems.

PRISM can be run from the command-line but it also offers a graphical user interface that can be seen in the picture bellow (Figure 5):

¹² PRISM, (2020), *Prism Model Checker*, <http://prismmodelchecker.org/>

¹³ KWIATKOWSKA M., NORMAN G., PARKER D., (2011), *PRISM 4.0: Verification of Probabilistic Real-time Systems*.

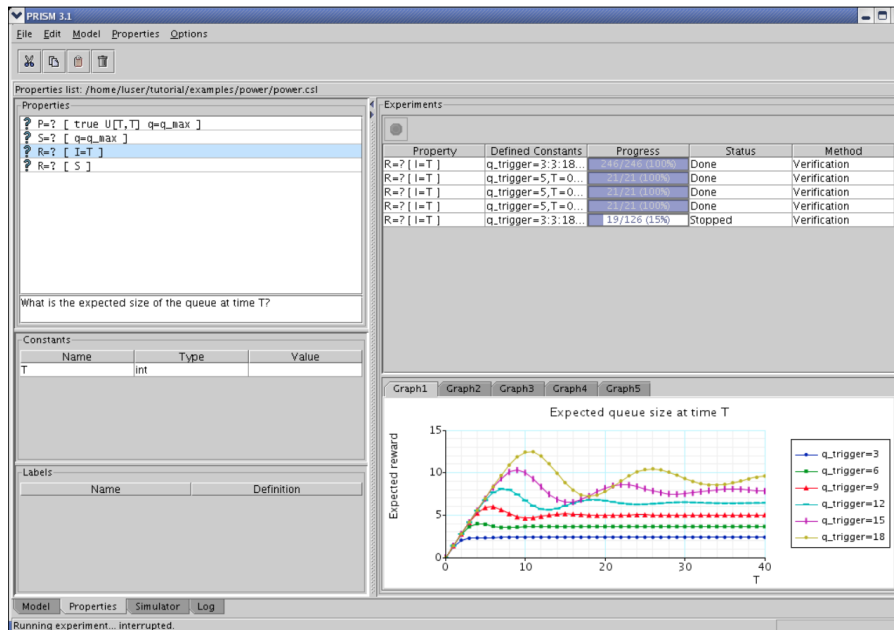


Figure 5: PRISM GUI

Source : <http://prismmodelchecker.org/screenshots.php>

vi. Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE)

SHARPE was first developed as a part of Robin Sahner's Ph.D. thesis in 1986¹⁴. It is a hierarchical modeling tool that analyzes stochastic models of reliability, availability, performance, and performability.

SHARPE allows the user to choose between several model types: fault trees, acyclic series-parallel graphs, reliability block diagrams, generalized stochastic Petri nets, acyclic Markov and semi-Markov models, cyclic Markov and semi-Markov models, and closed single- and multi-chain product-form queueing networks. For most of those model types, the users can choose among different analysis algorithm provided by SHARPE¹⁵.

SHARPE can be run from the command-line but it also offers a GUI that implements description techniques for most of the model types. This GUI that can be seen in the picture below (Figure 6):

¹⁴ TRIVEDI K. S., SAHNER R., (2009), *SHARPE at the age of twenty-two*, SIGMETRICS Perform.

¹⁵ Duke University SHARPE Portal, (2020), *SHARPE Portal*, <https://sharpe.pratt.duke.edu/>

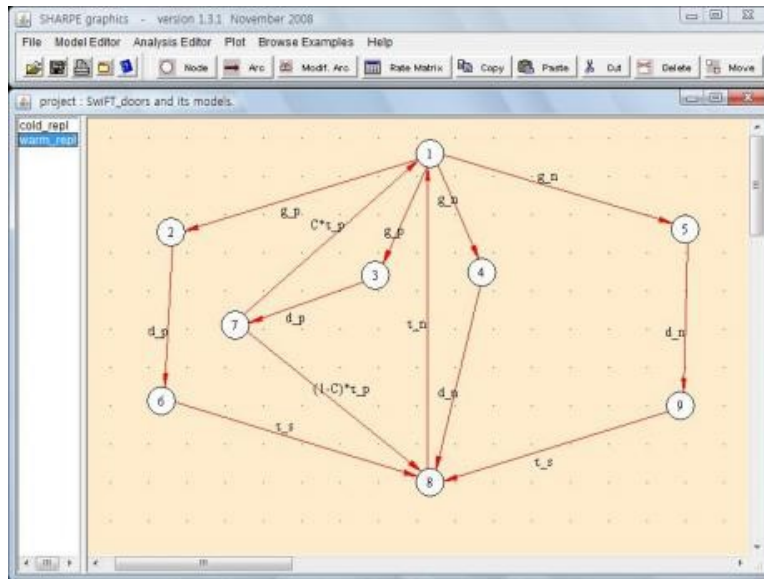


Figure 6: GUI of SHARPE

Source : <https://sharpe.pratt.duke.edu/system/files/Snapshot2.jpg>

vii. TANGRAM II

Tangram-II was developed for research and educational purposes. It provides an environment for system modeling and traffic measurement. Through the GUI, users can provide a high-level description of a system and obtain its performance analysis. This tool supports Markovian, non-Markovian models and Hybrid Markov Models (HMMs).¹⁶

The picture bellow (Figure 7) shows the GUI of TANGRAM II. On the left there is a Poisson source and, on the right, there is a Poisson server (with a queue).¹⁷

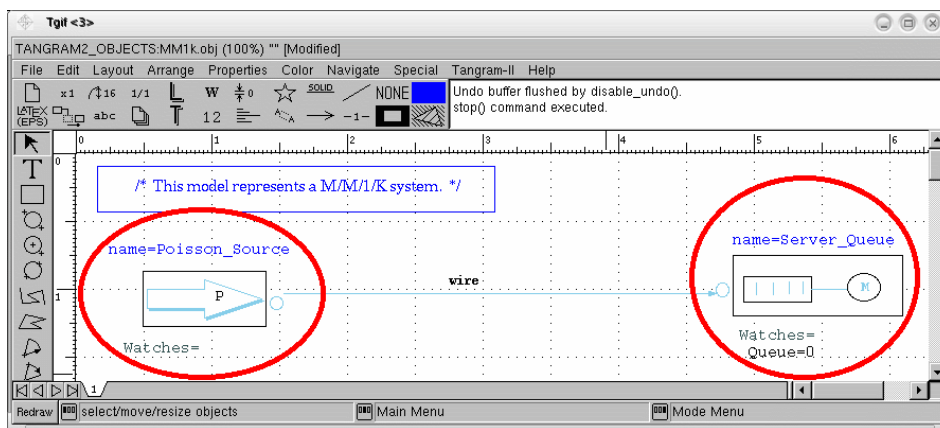


Figure 7: GUI of TANGRAM II

Source : http://www.land.ufri.br/tools/tangram2/tutorial/tangram2_firstSteps/figs/mm1k.gif

¹⁶ DE-SOUZA-E-SILVA E., FIGUEIREDO D. R., LEAO R. M.M., (2009), *The TANGRAM-II Integrated Modeling Environment for Computer Systems and Networks*

¹⁷ LAND (Laboratory for modeling, analysis and development of networks and computer systems), *Tangram – II*, <http://www.land.ufri.br/tools/tools.html>

III. Description of Java Modelling Tools (JMT)

Java Modelling Tools is a free open source tool suite for modelling computer and communication systems. It contains six tools: JSIMGraph, SIMwiz, JMVA, JABA, JWAT and JMCH. From 2006 to this day (26 June 2020), JMT was downloaded more than 70,500 times.¹⁸

The most recent release of JMT is the version 1.0.5 and it was released the 12th April 2020. The extensions presented in this document are based on this version.

In this section, Java Modeling Tools (JMT) will be presented in details

1. Brief history of JMT

Researchers at Politecnico were studying computer and communication models and, in order to make their work easily usable by others, they started the development of a software that would encapsulate their work within a more convenient and approachable interface.

In 1990, a first MSc thesis developed an MVA (mean value analysis) tool in C, then the programming language changed to C++.

In 2002, the code of the previously existing tool suite for performance evaluation called Windows modelling tools (WMT) was entirely converted to Java; and the JMT project started at Politecnico di Milano.

In 2013, the cooperation between Politecnico di Milano and Imperial College London started. Since then, many students and research assistants from Politecnico di Milano and from Imperial College were involved in the JMT project.

2. Description of JMT¹⁹

JMT is a suite of modelling tools that allows researchers and engineers to perform workload characterization, performance evaluation, capacity planning and modelling of computer and communication systems. The JMT suite can help software and system performance engineers to predict the behavior of a system and quickly answer what-if questions.

The JMT suite is composed by the six following tools (see Figure 8: Start screen of JMT):

¹⁸ CASALE G., SERAZZI G., BERTOLI M., (2020), *Java Modelling Tools- User manual*

¹⁹ CASALE G., SERAZZI G., (2011), *Quantitative System Evaluation with Java Modeling Tools (Tutorial Paper)*

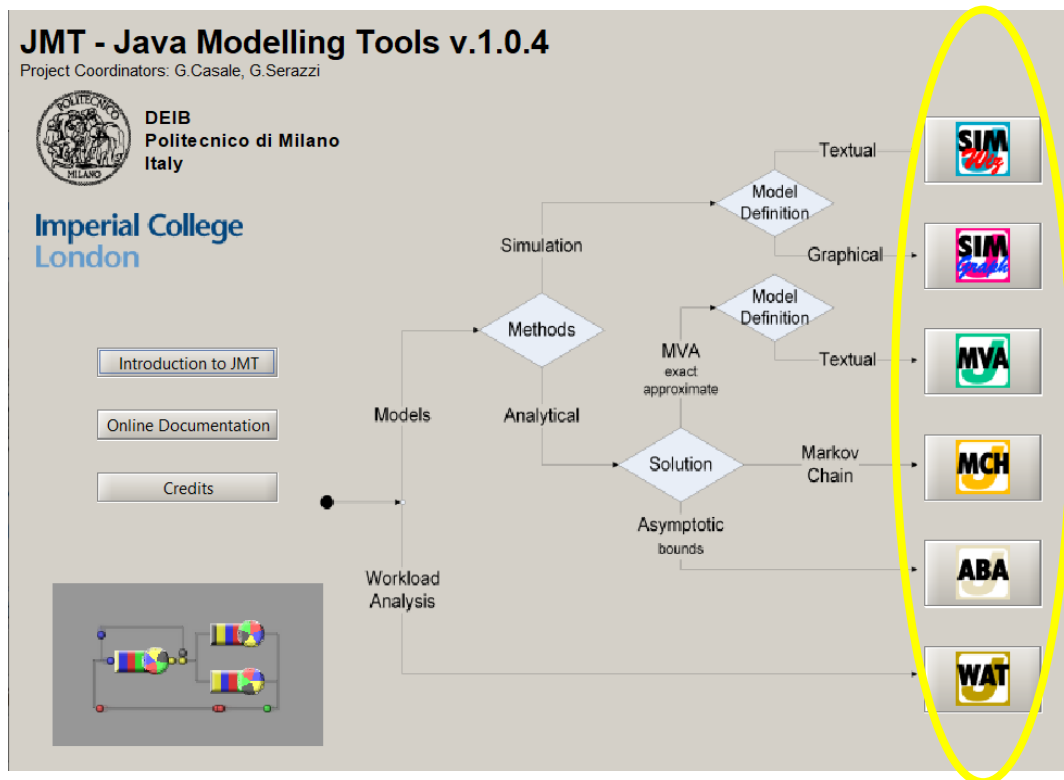


Figure 8: Start screen of JMT

i. [JSIMwiz](#)



JSIMwiz is a wizard-based interface for simulation of queueing network and Petri net models. With this tool, the user is helped in his process of defining the network by a sequence of wizard windows.

This tool supports state-independent routing strategies like Markovian and round robin; as well as state dependent routing strategies like load dependent routing, shortest response time with minimum queue length and server with minimum utilization.

The model build by the user can also contain fork-join servers (parallelism), place-transition structures (synchronization), class switching, priority classes and finite capacity regions (blocking).

JSIMwiz also supports What-if analyses, where a sequence of simulations is run for different values of control parameters.

ii. [JMVA](#)



With JMVA, the user specifies the structure of the model by textual wizards. The tool is based on a stabilized version of the Mean Value Analysis (MVA) algorithm and its extensions for single-class or multiclass product-form queueing networks, processing open, closed or mixed workloads.

It also provides What-if analyses and a graphical representation of the results.

This tool provides several exact algorithms for closed networks:

- MVA (Mean-Value Analysis),
- RECAL (Recursion by Chain Algorithm),

- MoM (Method of Moments),
- CoMoM (ClassOriented Method of Moments).

And several approximate solution algorithms:

- Chow,
- Bard-Schweitzer,
- AQL,
- Linearizer,
- De Souza-Muntz Linearizer

iii. [JABA](#)



JABA is an analytical tool for the automated identification of bottlenecks in multiclass closed product-form networks. The optimization studies (throughput maximization, identification of the optimal load, minimization of response time etc) can be performed thanks to the identification of the saturation sectors. Saturation sector are the mixes of requests of the different classes in execution that saturate several resources simultaneously.

This tool is able to analyze efficiently models with thousands of queues.

iv. [JWAT](#)



JWAT allows the workload characterization and provides algorithms for data scaling, k-means, fuzzy k-means clustering, outlier filtering and sample extraction for identifying similarities in the input data. The tool provides several input files formats (like Apache HTTP and IIS log files) but the users can also specify customized formats.

The clusters centroids represent the mean values of the parameters of the classes (e.g., CPU time, number of I/Os, number of web pages pages accessed) that can be used for the workload parameterization.

v. [JMCH](#)



JMCH is a graphical simulator that solves:

- single-station models, including single-server stations with finite (M/M/1/k) or infinite (M/M/1) queue sizes
- multiple-server stations with limited (M/M/c/k) or unlimited (M/M/c) queue sizes

The animation allows to visualize the simulation state both on the queue buffer and on a Markov model representing the system state

With this tool, the user can dynamically change the service time, the arrival rate and the queue size of the system.



JSIMGraph uses the same simulator engine as the one used by JSIMwiz, but instead of making the user define the network by a sequence of wizard windows, it provides a graphical user-friendly interface. This allows an easier description for both the network layout and the input parameters; and a nicer visualization of complex systems.

Moreover, JSIMGraph facilitates the process of performing an evaluation study by automatizing the critical statistical decisions, such as transient detection and removal, variance estimation, and simulation length control. This way, the users are not required to take decisions about parameters they may not be familiar with. Then, when all performance indexes are estimated with the required accuracy, the simulation automatically stops.

This tool supports what-if analyses, dynamical presentation of simulation state, performance metrics estimates and related confidence intervals. As JSIMwiz, it supports the evaluation of the most popular types of queueing models and several other constructs that cannot be solved by an exact analytical technique like non-product-form networks. JSIMGraph provides supports for fork-and-join elements, blocking mechanisms, burstiness, priorities, state-dependent routing strategies, user-defined general distributions, import and reuse of log data.

Arrival rates :

JSIMGraph generates arrival rates for open classes of customers by source stations and station service times (for any type of station in open and closed models). They can be generated according to the following distributions:

- Burst (General),
- Burst (MAP: Markovian Arrival Process),
- Burst (MMPP2: Markov-Modulated Poisson Process of order 2),
- Coxian,
- Deterministic,
- Erlang,
- Exponential,
- Gamma,
- Hyperexponential,
- Normal,
- Pareto,
- Phase-Type,
- Replayer,
- Uniform

Queueing disciplines:

JSIMGraph allows the use of preemptive and non-preemptive queueing disciplines:

- Non-preemptive:
 - First Come First Served (FCFS)
 - Random (RAND)
 - Last Come First Served (LCFS)
 - Longest Job First (LJF)
 - Shortest Job First (SJF)
 - Longest Expected Processing Time (LEPT)
 - Shortest Expected Processing Processing Time (SEPT)
- Preemptive:
 - Processor Sharing (PS)
 - Discriminatory Processor Sharing (DPS)
 - Generalized Processor Sharing (GPS)

In case of limited capacity regions (blocking regions), the available drop rules are: Drop, Waiting Queue and Blocking After Service (BAS).

Routing strategies

The path followed by requests among the resources can be described probabilistically or by using one of the following routing strategies:

- Fastest Service
- Least Utilization
- Load Dependent Routing
- Random
- Join the Shortest Queue
- Shortest Response Time
- Round Robin (with and without weights)
- Power of k choices (with and without memory)
- Disabled (the values of the control parameters are evaluated on the stations connected in output to the considered one)

These strategies can be further combined among themselves through the use of a routing station.

3. Study case using JSIMGraph

In this section, two study cases will be presented to show the kind of problems that JSIMGraph can solve. The study cases are taken from the JMT user manual (CASALE G., SERAZZI G., BERTOLI M., (2020), *Java Modelling Tools- User manual*)

i. First infrastructure: A computing server and a storage server equipped with two disks²⁰

In this scenario, we study the digital infrastructure of a company, the goal is to evaluate the performances of the system, in particular we want to evaluate the system throughput and response time. The infrastructure is composed of a computing server and a storage server equipped with two disks.

The workload is considered as a single closed class with a population of $N = 3$ customers.

The infrastructure is composed of four stations: the computing server and the disks are modeled by queueing load independent stations (CPU, Disk1 and Disk2), and the user's think time ($Z=16s$) between interactions with the system is modeled by a delay center (users). The following image (Figure 9) shows the infrastructure layout (done using JSIMGraph):

²⁰ CASALE G., SERAZZI G., BERTOLI M., (2020), *Java Modelling Tools- User manual*, Chapter 3, page 104

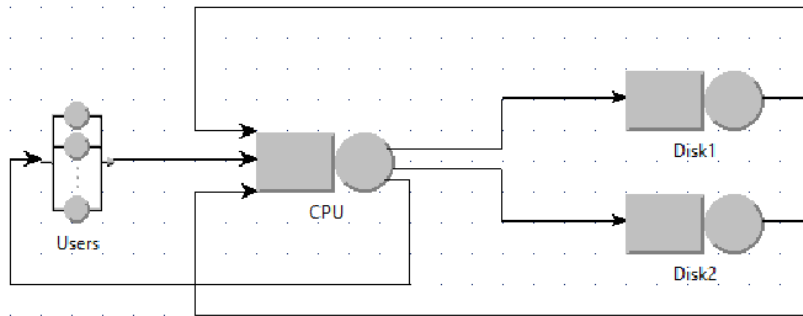


Figure 9: Network layout, first study case

The service times are distributed exponentially. Their value per station, together with the values of the Visits are shown in the following table (Table 1):

	CPU	Disk1	Disk2	Users
Service time (sec)	0.006	0.038	0.030	16
Visits	101	60	40	1

Table 1: Values for Service time and visits, first study case

The model is parametrized with the following parameters (Table 2):

Station	Service distribution	Routing strategy
Users	Exponential $\lambda=16$	Random
CPU	Exponential $\lambda=0.06$	Probabilities: - Disk1: 60/101 - Disk2: 40/101 - Users: 1/101
Disk1	Exponential $\lambda=0.038$	Probabilities: - CPU: 1
Disk2	Exponential $\lambda=0.030$	Probabilities: - CPU: 1

Table 2: Stations parameters, first study case

After parametrizing the model, we can choose the performance indices that interest us. The selected performance indices will be collected and plotted by the simulation engine.

We select the following performance indices:

- Throughput [jobs/sec] (for CPU, Disk1, Disk2, Users and the System)
- Number of customers [jobs] (for CPU, Disk1, Disk2, Users and the System)
- Residence time [sec] (for CPU, Disk1, Disk2, Users)
- Utilization (for CPU, Disk1, Disk2 and Users)
- Response time [sec] (for the system)

When running the simulation, JSIMGraph colors the stations according to the percentages of station busy time and of the queue length due to customers (see Figure 10). JSIMGraph also displays, the progress of the average values of the selected performance indices and the extremes of the confidence intervals. All the results can then be exported in csv format.

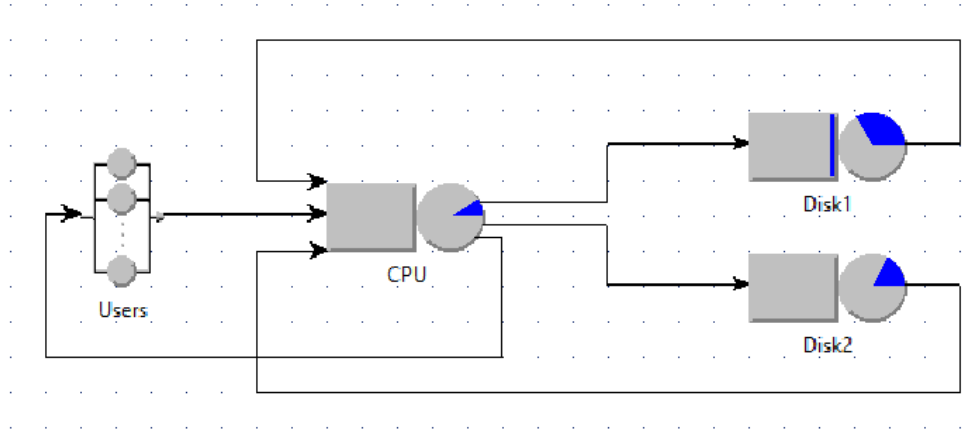


Figure 10: Network layout after simulation, first study case

Table 3 shows the average, min and max values of the performance indices at the station level at the and at the level of the global system.

		Residence time [sec]	Utilization [%]	Throughput [jobs/sec]	Response Time [sec]	Number of customers [jobs]
CPU	Min	0.8915	0.0788	13.6539		0.0840
	Max	0.9843	0.0950	16.4400		0.0988
	Avg	0.9379	0.0869	14.9179		0.0914
Users	Min	14.9246	2.1573	0.1391		2.1573
	Max	16.3191	2.4757	0.1573		2.4757
	Avg	15.6218	2.3165	0.1476		2.3165
Disk1	Min	3.0748	0.2993	7.9733		0.3811
	Max	3.7180	0.3576	9.1456		0.4537
	Avg	3.3964	0.3285	8.5193		0.4174
Disk2	Min	1.6924	0.1622	5.3243		0.1779
	Max	1.9179	0.1887	6.2704		0.2093
	Avg	1.8052	0.1755	5.7587		0.1936
System	Min			0.1387	19.1441	-
	Max			0.1575	21.7909	-
	Avg			0.1475	20.4675	3.000

Table 3: Simulation results, first study case

This analysis of the digital infrastructure of the company, allowed us to quickly and easily evaluate its performances at the scale of the system but also as the scale the station.

ii. Second infrastructure: Study of the behavior of a station after a growth of service time²¹

In this second scenario, we study the infrastructure of an Intranet composed of a storage server and an Apache server. The storage server consists of three disks (modeled by queuing stations) accessed according to a RAID0 striping algorithm (modeled by a fork station). The Apache web server is composed of two independent servers, preceded by a load balancer that splits the request among them. For performance reasons, the Apache web server can only accept a limited number of concurrent requests. The Apache server is therefore modelled as a Finite Capacity Region with the maximum number of customers MaxClients = 100.

The figure below (Figure 11) shows the layout of the infrastructure

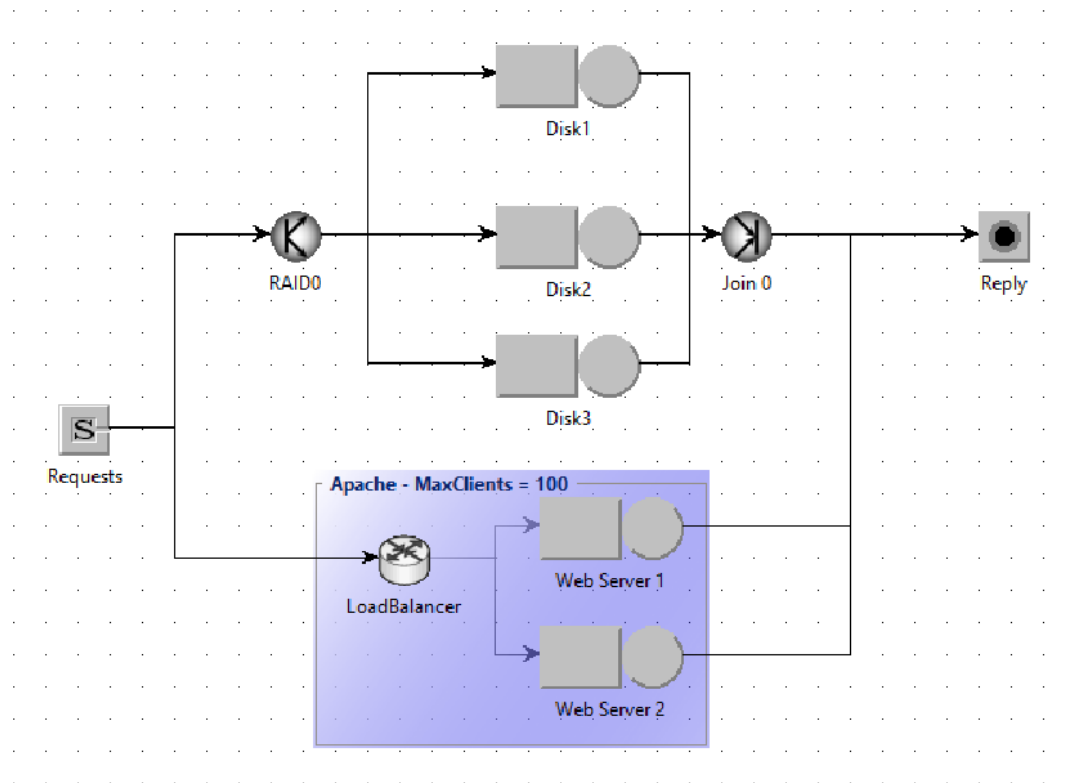


Figure 11: Network layout, second study case

The goal of the analysis we will perform is to predict the behavior of the first web server (station Web Server 1) if its service time grows by 150%

The customers are subdivided into two open classes, Class0 and Class1. The customers of class0 are only allowed to access the Apache server while the customers of class1 can visit either the storage or the Apache server according to a probabilistic algorithm.

²¹ CASALE G., SERAZZI G., BERTOLI M., (2020), *Java Modelling Tools- User manual*, Chapter 3, page 107

Interarrival times of class0 and class1 customers follows the distribution below:

- **Class0:** hyperexponential with parameters $(p, \lambda_1, \lambda_2) = (0.291, 0.182, 0.442)$,
- **Class1:** exponential with $\lambda=5$

The RAID0 striping algorithm is modeled by a Fork station. When a request arrives at station RAID0, it is splitted in three requests sent in parallel to the three disks of the storage server. The Fork degree at RAID0 station is 3. This means that for each incoming request, three requests are generated in each outgoing link.

The following table (Table 4) shows the service distribution and Routing strategy of the stations of the network.

Station		Service distribution	Routing strategy
Requests	Class0	/	Probabilities: - LoadBalancer: 1
	Class1		Probabilities: - LoadBalancer: 0.5 - RAID0: 0.5
Disk 1	Class0	Exponential $\lambda=5$	Random
	Class1	Exponential $\lambda=0.5$	Random
Disk 2	Class0	Exponential $\lambda=5$	Random
	Class1	Exponential $\lambda=0.5$	Random
Disk 3	Class0	Exponential $\lambda=5$	Random
	Class1	Exponential $\lambda=0.5$	Random
LoadBalancer	Class0	/	Shortest Queue Length
	Class1		Shortest Response Time
Web Server 1	Class0	Exponential $\lambda=1$	Random
	Class1	Exponential $\lambda=1$	Random
Web Server 2	Class0	Exponential $\lambda=1$	Random
	Class1	Exponential $\lambda=1$	Random

Table 4: Stations parameters, second study case

In this scenario, JSIMGraph is used to study the behavior of the first web server (station Web Server 1) if its service time for all classes grows by 150%: This is a *What-If* analysis. Particularly, the company owning the network needs to predict the System Response Time for both Class0 and Class1 customers at confidence interval 0.99 and maximum relative error 0.05. It is also needed to know the global throughput of WebServer1 at confidence interval 0.90 and maximum relative error 0.10.

Therefore, we select the following performance indices

- Throughput [jobs/sec] (for all classes for Web Server 1)
 - Response time of the system [sec] (for class0 and class1)
- **Throughput of station Web Server 1 (for all classes)**

The results of the What-if analysis in terms of Throughput for both Class0 and Class1 customers is shown by Figure 12 and Table 5.

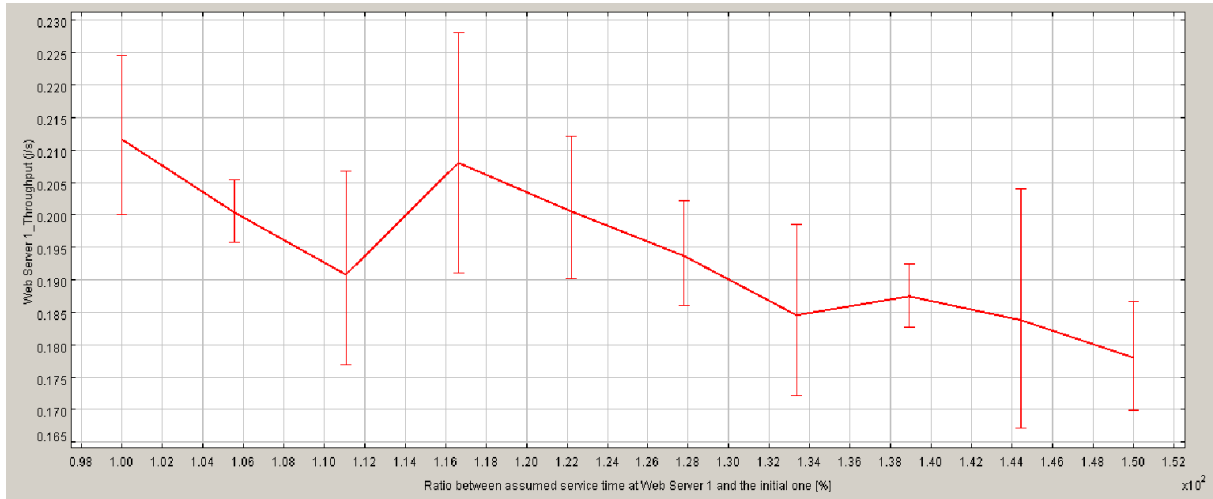


Figure 12: Results on throughput (all classes), second study case

	100.00%	105.56%	111.11%	116.67	122.22%	127.78%	133.33%	138.89%	144.44%	150.00%
Mean Value [jobs/s]	0.2116	0.2004	0.1907	0.2080	0.2005	0.1937	0.1844	0.1874	0.1837	0.1779
Max [jobs/s]	0.2246	0.2053	0.2068	0.2281	0.2121	0.2021	0.1985	0.1924	0.2040	0.1867
Min [jobs/s]	0.2000	0.1957	0.1769	0.1911	0.1901	0.1860	0.1722	0.1827	0.1671	0.1700

Table 5: Results on throughput (all classes), second study case

- **Response time of the system for Class0**

The results of the What-if analysis in terms of System Response Time for Class0 customers is shown by Figure 13 and Table 6.

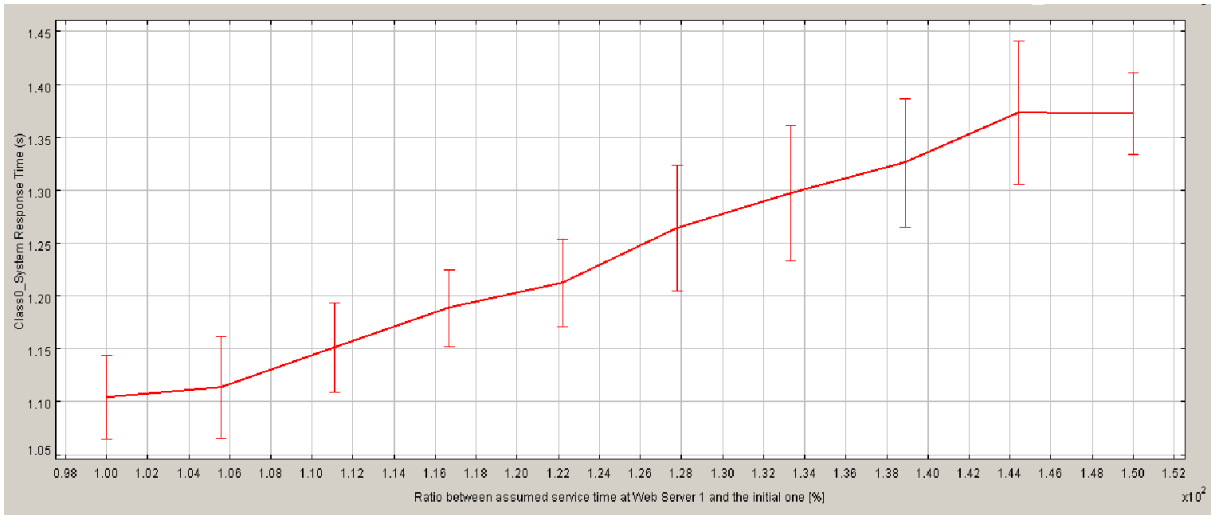


Figure 13: Results on system response time for Class0, second study case

	100.00%	105.56%	111.11%	116.67	122.22%	127.78%	133.33%	138.89%	144.44%	150.00%
Mean Value [jobs/s]	1.1040	1.1137	1.1515	1.1884	1.2123	1.2644	1.2974	1.3257	1.3732	1.3723
Max [jobs/s]	1.1438	1.1619	1.1940	1.2245	1.2541	1.3238	1.3613	1.3868	1.4414	1.4106
Min [jobs/s]	1.0643	1.0655	1.1090	1.1523	1.1706	1.2050	1.2335	1.2647	1.3051	1.3339

Table 6: Results on system response time for Class0, second study case

- **Response time of the system for Class1**

The results of the What-if analysis in terms of System Response Time for Class1 customers is shown by Figure 14 and Table 7: Results on system response time class1, second study case Table 7.

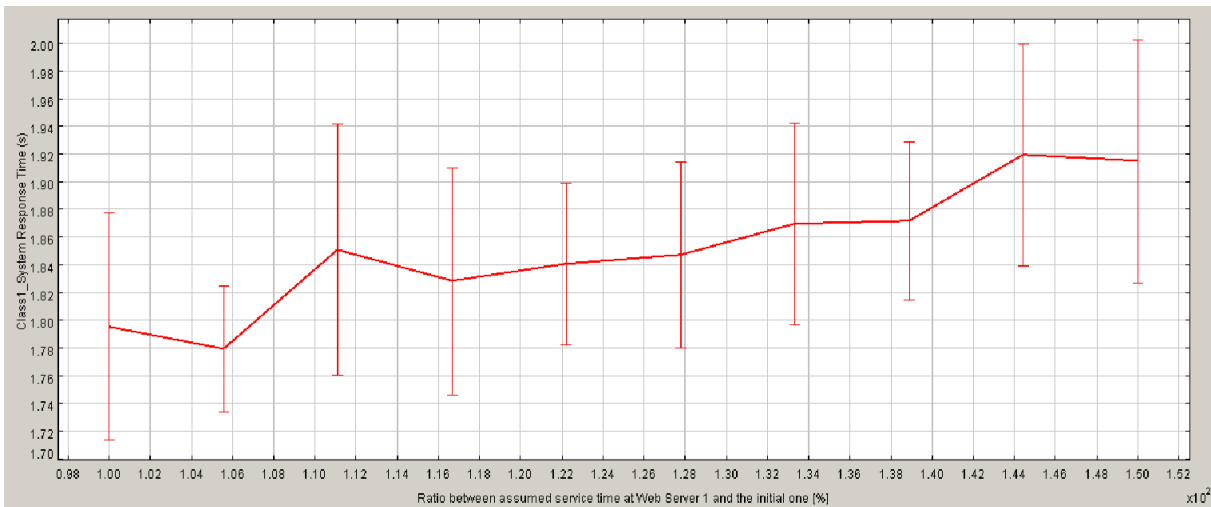


Figure 14: Results on system response time class1, second study case

	100.00%	105.56%	111.11%	116.67	122.22%	127.78%	133.33%	138.89%	144.44%	150.00%
Mean Value [jobs/s]	1.7951	1.7795	1.8509	1.8280	1.8406	1.8472	1.8695	1.8718	1.9192	1.9150
Max [jobs/s]	1.8771	1.8250	1.9414	1.9103	1.8992	1.9142	1.9423	1.9287	1.9996	2.0027
Min [jobs/s]	1.7132	1.7340	1.7603	1.7458	1.7820	1.7802	1.7967	1.8148	1.8389	1.8272

Table 7: Results on system response time class1, second study case

IV. Improving JSIMGraph

As we saw in the previous section, JSIMGraph is a powerful tool for performing the analysis of computer systems and predicting their behaviors, but according to feedbacks from the users of JMT, some aspects of the graphical user interface could still be improved.

1. The graphical user interface of JSIMGraph

The graphical user interface of JMT is built with Java Swing: a lightweight GUI widget toolkit used to create windows-based applications. It is entirely written in Java and thus is platform-independent. Java Swing is part of the Java Foundation Classes (JFC) and is built on the top of the Abstract Windowing Toolkit (AWT) API.


Inside tool JSIMGraph, the network model is drawn using the class Graphics2D of Swing.

2. Aspects of JSIMGraph that could be improved

This section describes some of the issues concerning the GUI faced by the users of JMT. This section, as well as the rest of this document, focuses on the version 1.0.5 of JMT.

i. The automatic connection shapes generating a confusing layout

When a user wants to connect two stations together, he performs the following steps:

1. Clicks on the connection button: 
2. Clicks on the source station
3. Drags the mouse to the target station
4. Releases mouse on the target station

Then, the shape of the arc is generated automatically.

The pro of this solution is that the user does not have to build the connection shape himself so he saves time. The con is that sometimes, the automatic shape is not the one the user would like to have. When a very complex system is modeled, the cons exceed the pros. Indeed, the automatic arcs shapes can

sometime make the graph difficult to understand. In that case the only possibility for the user to improve the layout is to move the stations and hope that the new automatically generated arc will be nicer.

The figure below (Figure 15) shows the model of a rock-collapse forecasting system. More details about the system can be found in the publication *Capacity Planning of Fog Computing Infrastructures for Smart Monitoring* written by Riccardo Pincioli, Marco Gribaudo, Manuel Roveri and Giuseppe Serazzi²².

This network is given as an example in this section because its layout suffers from unclear connections. Especially the arcs from Station Transmitting to station Deplete, from Station Deplete to station Transmitting and from Station Transmitting to station Rest are not easily distinguishable.

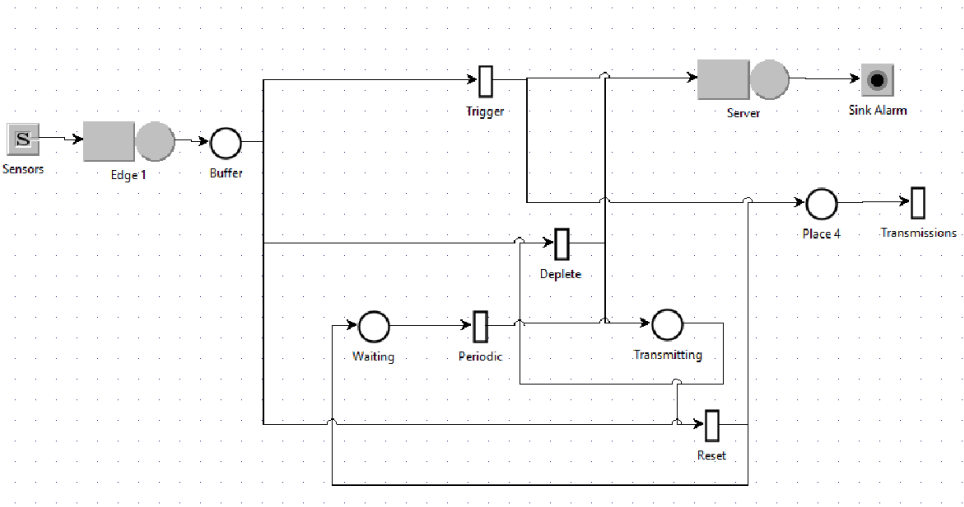


Figure 15: Model with problematic arc shapes

ii. The input and output overlap

Another reason for which the connections are hardly readable is that when several arcs connect the same stations, their shape overlap at the ports of the stations.

In Figure 16, we can see that two arcs are existing from station Deplete and that two arcs are entering station Transmitting. But in input of the station Transmitting, the two arcs coincide, so we see only one arrow. At the output of station Deplete where two arcs exit, we observe the same phenomenon: the two arcs completely overlap.

²² PINCIROLI R., GRIBAUDO M., ROVERI M., SERAZZI G. (2018) *Capacity Planning of Fog Computing Infrastructures for Smart Monitoring*

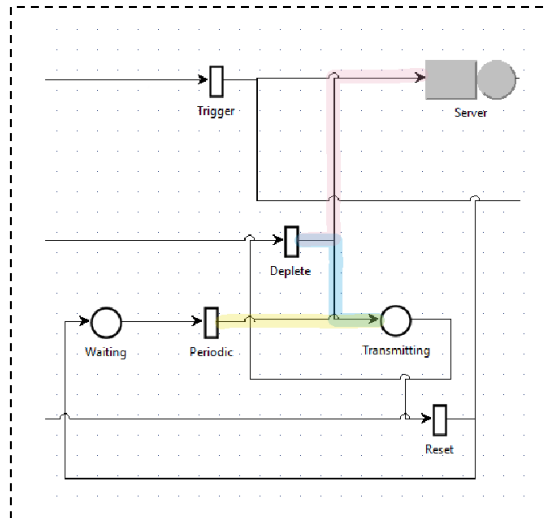


Figure 16: highlight of port input and output overlap problem

iii. The impossibility of quickly deleting an arc.

In JSIMGraph version 1.05, the arcs are non-selectable. So, if a user makes a mistake when connecting two stations, he does not have the possibility to simply select and delete the wrong arc. In order to do so he must delete the source or target station of the arc. This can cause a non-negligible waste of time, especially if the station simulation parameters were already filled or if the station had other connections.

iv. The impossibility to rotate stations

The station can be mirrored vertically as shown in Figure 17 but they can't be rotated.



Figure 17: Example of station being mirrored

This is problematic when the system being modeled has the kind of layout shown in Figure 18.

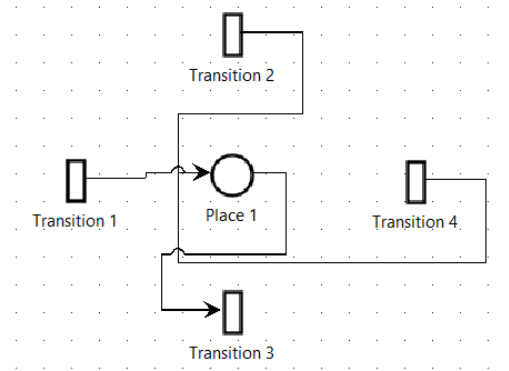


Figure 18: Example of problem caused by the impossibility to rotate stations

With this type of network architecture, it would be preferable have the possibility to rotate the stations Transition 3, Transition 4 and Place1 by 90°. The following figure (Figure 19) shows how much the rotation of stations could improve the layout in that case.

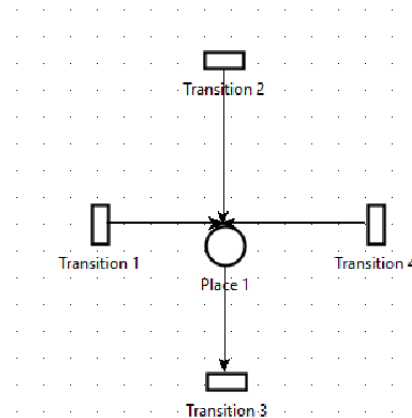


Figure 19: Example of a nicer layout that could be obtained by rotating stations

v. The difficulty to properly and quickly align stations

The current version of JSIMGraph does not give the possibility to the users to snap stations to a grid. The consequence is that it can be inconvenient and time consuming for the users to correctly align stations without any guidance. And, if connected stations are not correctly aligned, the shape of the connection will present a little undesirable step.

3. Proposed improvements

The goal of my thesis at Politecnico di Milano was to solve the problems mentioned before. In order to do that, the following improvements were implemented:

- I 1. Give the possibility to the user to choose between two types of connections with a different graphical appearance but with the same semantic meaning. The first type of connection is the one already existing (with the previously described pros and cons). The second type of connection allows the user to draw the shape of the arc with the mouse. With the new connection type, the user has the possibility to choose the intermediary points of the arc and to do sharp or curved edges (thus solving the problem presented in section “The automatic connection shapes”).
- I 2. Give the possibility to the user to modify the shape of a hand draw connection once it has been created. This implies the possibility to select arcs (thus solving the problem presented in section “The impossibility of quickly deleting an arc”) and the possibility to move the input and output of arcs (solving the problem presented in section “The input and output overlap”).
- I 3. Allow the rotation of stations. (cf section “The impossibility to rotate stations”)
- I 4. Give the possibility to the user to snap items (stations and arcs) to the grid (cf section “The difficulty to properly and quickly align stations”).

4. Requirements

The Table 9 and Table 8 describe more precisely the extensions that were developed during this thesis work.

NON-FUNCTIONAL REQUIREMENTS	
Requirement id	Description
NR1	The new version of the tool must retro compatible
NR2	The development language must be JAVA
NR3	The GUI extensions must use Swing
NR4	The tool must able to open models created with previous versions of JMT

Table 8: non-functional requirements

FUNCTIONAL REQUIREMENTS		
Requirement id	Description	
I1	R1	The system must offer the possibility to connect stations with a Bezier connection or an automatic connection
	R2	The system must allow the user to draw Bezier connections using the mouse
	R2.1	The system must allow the user to draw arcs with an arbitrary number of intermediary points
	R2.2	The system must allow the user to draw arcs with curved or sharp edges
I2	R3	The system must offer the possibility to replace a connection of one type by a connection of the other type (Bezier / automatic)
	R4	The system must allow the selection of connections
	R5	The system must allow the modification of connection of Bezier type
	R5.1	When a Bezier connection is selected, the system must make visible to the user the control points responsible for the shape of the connections
	R5.2	The system must allow the user to move the control points of a Bezier connection
	R5.3	The system must allow the user to add control points to a Bezier connection
	R5.4	The system must allow the user to remove control points from a Bezier connection
	R5.5	The system must allow the transformation of sharp edges into curved edges (and inversely) for Bezier connections.
R5.6	The system must allow the user to break the continuity of a connection (this means that the user creates a hole in the connection)	
I3	R6	The system must allow the rotation of station by hops of 45° to the left or to the right
I4	R7	The system must allow to snap stations and arcs to a grid
	R7.1	The system must allow the user to enable and disable the grid functionality as he wishes.

Table 9: functional requirements

V. Implementation of the extensions

1. Modelling of Bezier curves

The objective is to give the users the possibility to connect stations by drawing arbitrary shapes with the mouse. The user must be able to draw shapes with sharp or curved edges and with an arbitrary number of intermediary points.

The figure below (Figure 20) shows an example of the type of shape that is made possible with this new feature:



Figure 20: Example of new connection shape

The shapes of the connection are implemented using Bezier curves

i. Definition of Bezier curves

A Bezier curve is a parametric curve related to Bernstein polynomial. Bezier curves are named after Pierre Bezier, who used it in the 1960s for designing curves for the bodywork of Renault cars.

General definition

The Bezier curve for the $n+1$ controls points (P_0, \dots, P_n) is the set of points defined by the parametric representation: $P(t) = \sum_{i=0}^n B_i^n(t) * P_i$ with $t \in [0 ; 1]$ and with B_i^n the Bernstein polynomial :

$$P(t) = \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i P_i$$
$$= (1-t)^n P_0 + \binom{n}{1} (1-t)^{n-1} t P_1 + \dots + \binom{n}{n-1} (1-t) t^{n-1} P_{n-1} + t^n P_n \quad 0 \leq t \leq 1$$

The series of points P_0, \dots, P_n forms the *Bezier polygon* (or *control polygon*). The convex hull of the Bezier polygon contains the Bezier curve.

To implement the new type of connection in JSIMGraph, a specific type of Bezier curves is used: Cubic Bezier curves.

Specific case: Cubic Bezier curves

A cubic Bezier curve is defined by four points P0, P1, P2 and P3. The curve starts from P0 and is directed towards P1 and arrives at P3 coming from the direction of P2. In general, the curves don't intersect with the points P1 and P2: those points only give an indication on direction. The figure below (Figure 21) shows an example of a cubic Bezier curve:

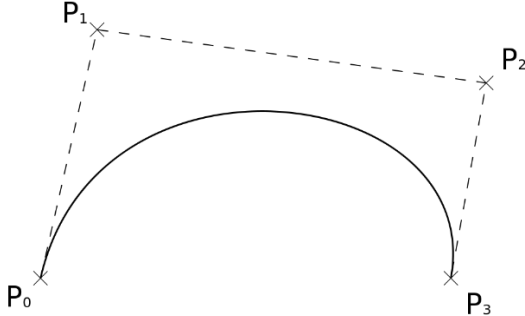


Figure 21: Example of cubic Bezier curve

A cubic Bezier curve can be defined as an affine combination of the four controls points. Its explicit expression is:

$$P(t) = (1 - t)^3 P_0 + 3(1 - t)^2 t P_1 + 3(1 - t) t^2 P_2 + t^3 P_3 \quad \text{with } 0 \leq t \leq 1$$

ii. The implementation

The shape of a connection is built using a sequence of cubic Bezier curves. The Figure 22 shows an example of a connection composed of three cubic Bezier curves.

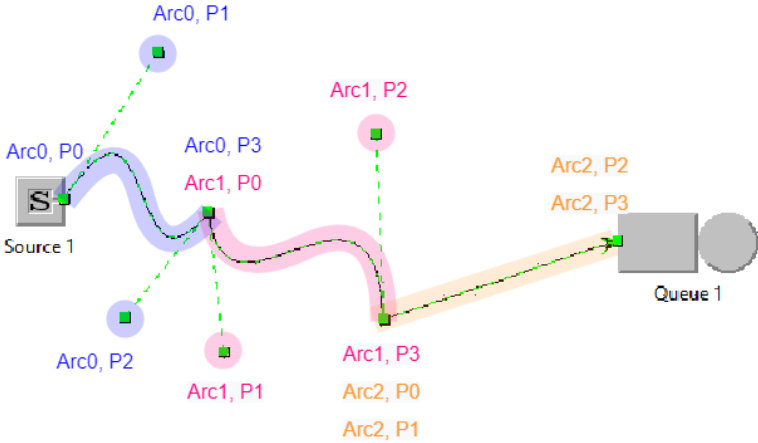


Figure 22: Example of structure of new connection shape

As observed in Figure 22, a straight line is obtained by making the controls points P1 and P2 coincide respectively with P0 and P3.

As java is an oriented-object language, two new classes were created to represent a connection shape.

- **Class *JMTPath*:**
Represents the shape of the connection. It has an attribute *arcs* which is a list of instances of **JMTArc**

- **Class *JMTArc*:**
Represents the shape of a portion of connection. A JMTArc can be written as a cubic Bezier curve. It has three attributes:
 - *source*: a point representing the position of the source the arc (called P0 in the Figure 22)
 - *arcPoints*: a list of points representing the positions of the control points of the arcs (called P1 and P2 in the Figure 22)
 - *target*: a point representing the position of the target the arc (called P3 in the Figure 22)

Before the implementation of this new feature, the connection shapes were built automatically by the software. The construction of the connection was done dynamically so there was no need to store data about the shape of the connection. Now the user also has the possibility to draw by himself the connections shapes, this implies that the shape of the connection needs to be stored.

When two stations are linked by a Bezier connection, the shape of that connection is stored in the model in the form of a hash map where the shape of the connection (an instance of JMTPath) can be retrieved from the identifiers of the source and target stations.

The position of the points of the shape are relative to the position of the stations following the following schema:

For all arcs except the last:

- Point P0: relative to the position of the source station
- Point P1: relative to P0
- Point P2: relative to P3
- Point P3: relative to the position of the source station

For the last arc:

- Point P0: relative to the position of the source station
- Point P1: relative to P0
- Point P2: relative to P3
- Point P3: relative to the position of the target station

The points P1 and P2 indicate the direction of the arc at points P0 and P3 respectively, they will therefore be called tangents.

iii. [New format for the xml file](#)

When the user presses the save button, the connections shapes stored in the model are written in the xml file. The xml file is the file that contains all the information about a model (the stations, their

positions, the connections, the parameters (queue strategies, routing strategies...), the simulation results etc). In order to be able to display the correct connections shapes when a model is opened, the xml file must also contain the shapes of the connections for stations connected with Bezier connections.

The following extract of the XML file shows how the connection shapes are defined for the example shown in Figure 22.

```
.  
. .  
. .  
    <connectionShape source="Source 1" target="Queue 1">  
        <arc>  
            <source x="0.0" y="0.0"/>  
            <point x="59.82499999999999" y="-93.68799999999999"/>  
            <point x="-53.0" y="67.0"/>  
            <target x="91.82499999999999" y="10.312000000000012"/>  
        </arc>  
        <arc>  
            <source x="91.82499999999999" y="10.312000000000012"/>  
            <point x="32.82499999999999" y="87.31200000000001"/>  
            <point x="-42.0" y="-90.0"/>  
            <target x="208.825" y="30.312000000000012"/>  
        </arc>  
        <arc>  
            <source x="208.825" y="30.312000000000012"/>  
            <point x="0.0" y="0.0"/>  
            <point x="0.0" y="0.0"/>  
            <target x="0.0" y="0.3400000000000034"/>  
        </arc>  
    </connectionShape>  
. . .
```

This new version of JMT supports both types of connections shape: automatic and Bezier.

When a file is opened in JMT, the first step is to read the XML file. If a connection shape is found in the XML file, and if the model contains a corresponding connection for that connection shape, then the connection shape is added in the model in the form of a hash map that links the shape of the connection with the identifiers of the source and target stations.

Then, when the arcs need to be rendered, the renderer checks if a connection shape exist in the model, if yes, the connection is rendered according to the specified shape, otherwise the connection is rendered by dynamically computing a shape.

So, if a model created with an older version is opened with this new version of JMT, the XML file will not contain any connections shapes. So, the connections will be rendered using the automatic shape computation: the result will be the same as if the model was opened in the previous version of JMT. Therefore, there is no compatibility issue between the two versions of JMT. After opening a model made with the older version of JMT, the user will have the possibility to change the connection types by transforming any existing connection into a Bezier connection.

If a model containing connections shapes is opened with the older version of JMT, the section of the XML file concerning the connection shape will simply be ignored, so the connections will be displayed as if they were automatic connections.

2. The process of drawing of Bezier connections.

For connecting stations, the user has the choice between an automatic edge or an edge that he can draw himself. The JSIMGraph tool has two buttons for connecting stations:



For connecting stations with an automatic connection

For connecting stations with a manual Bezier connection

When a user wants to connect two stations together with a Bezier connection, he performs the following steps:

1. Clicks on the connection button:
2. Clicks on the source station
 - If the shift key was pressed when doing the click:*
 - 2.1 drags the mouse to select the position of the control point P1
 - Otherwise the arc is sharp (P1 is set at the same location as P0)*
3. Moves the mouse to the position of the intermediary point (Point P3 of the current arc and point P0 of the successive arc) and clicks
 - If the shift key was pressed when doing the click:*
 - 3.1 drags the mouse to select the position of the control point P2 of the current arc and of the control point P1 of the successive arc
 - Otherwise the arc is sharp (P2 is set at the same location as P3)*
4. Repeats the step 3 for how many intermediary points necessary
5. Moves the mouse to the location of the target station and clicks
 - If the shift key was pressed when doing the click:*
 - 5.1 drags the mouse to select the position of the control point P2
 - Otherwise the arc is sharp (P2 is set at the same location as P3)*

At any time, the user can stop the process of drawing the arc by pressing the key Escape on his keyboard.

This process can also be explained as a state diagram:

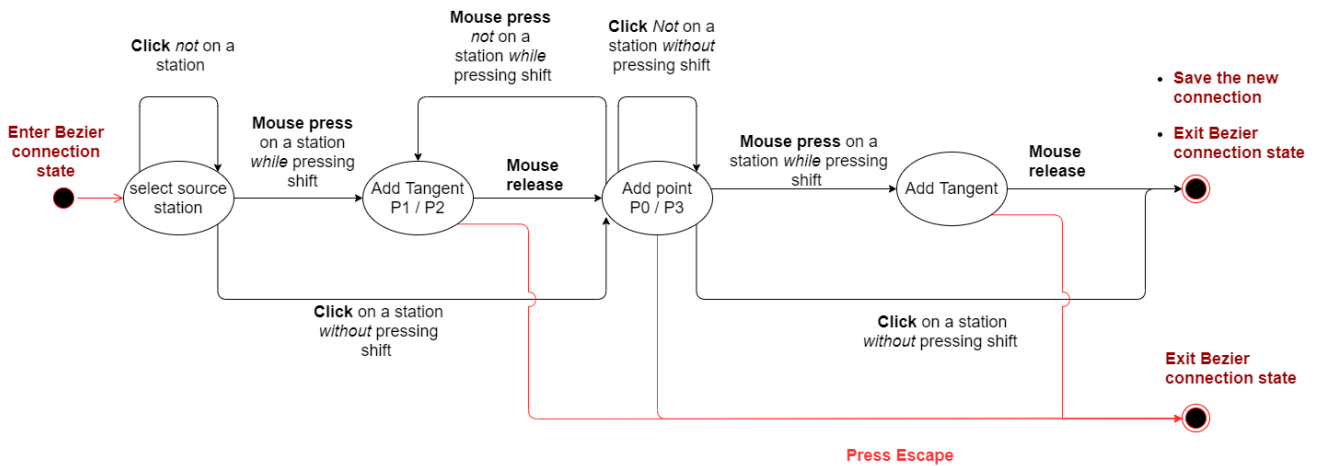


Figure 23: State diagram of the process of drawing Bezier connections

In order to facilitate the drawing phase, and because this is not restrictive in most cases, the users only have the possibility to create connections that respect the following rules:

- The connection is continuous:
If the connection contains n arcs and $n \geq 2$, then $\forall i \in [0, n - 1], P3_{Arc\ i} = P0_{Arc\ i+1}$
- The tangents are symmetric:
If the connection contains n arcs and $n \geq 2$, then $\forall i \in [0, n - 1], P2_{Arc\ i} = -P1_{Arc\ i+1}$

The following figure (Figure 24) presents a visual example:

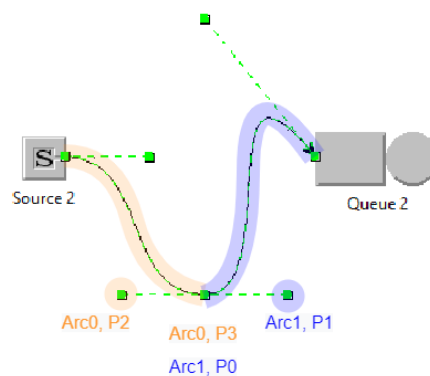



Figure 24: Example of a Bezier connection showing the continuity and symmetry of tangents

Once the connection has been created, the user has the possibility to modify the shape of the connection so that it doesn't respect the conditions of continuity and of symmetry of the tangents.

3. Modifying the connection shapes

Once an arc has been created, it can be deleted and its shape can be modified.

In order to delete an arc, the user simply has to select the arc by clicking on it. Once the arc is selected, it can be deleted the same way as the stations: by clicking on the button  or by pressing Del on the keyboard.

The arc selection feature has been implemented for both the new connection type and the old one (the automatic one).

When the user selects a Bezier connection a tool box appears presenting buttons for all the possible actions that can be done on an arc:

- Moving control points
- Adding intermediary points
- Deleting control points
- Adding tangents to sharp control points
- Breaking the tangent symmetry
- Breaking the continuity

i. Moving controls points

The user can move any type of control point: tangents points (P1 and P2) as in Figure 25, and also intermediary points like P0 and P3. The points can be moved anywhere on the canvas, except for the point P0 of the first arc and the point P3 of the last arc. For those points, the user is allowed to move the point only inside the boundary of the station. The authorized zone is highlighted by a rectangle as seen in Figure 26.

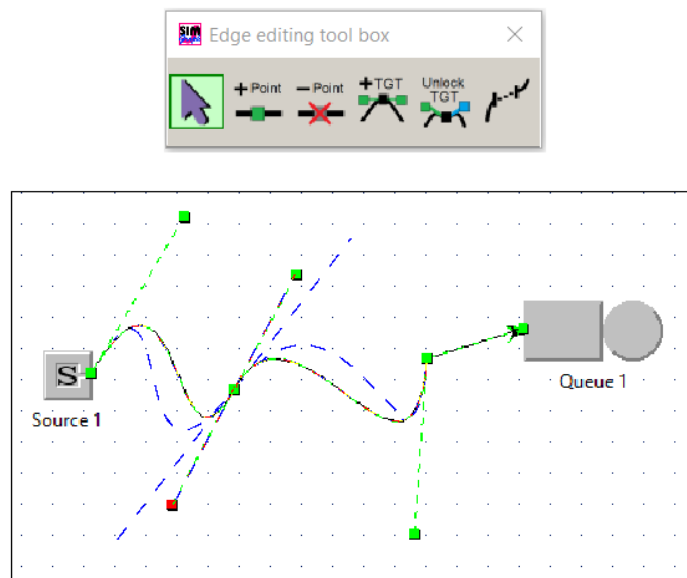


Figure 25: moving point P2 of the first arc

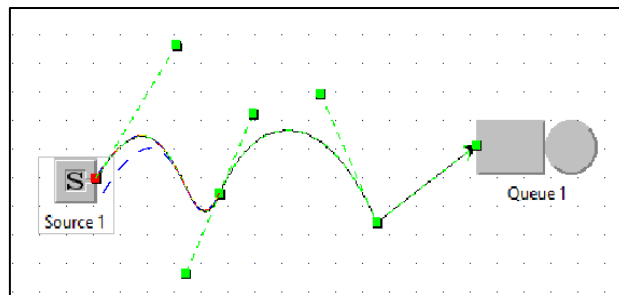


Figure 26: Moving point P0 of the first arc

As seen in Figure 24, the tangents (Points of P1 and P2) are symmetric by default. This means that moving a point P2 also implies moving the point P1 of the next arc in order to keep the symmetry. If the user wants to move tangents independently, he has to press the fifth button and performs the action described in section “Breaking the tangent symmetry”.

The process of moving control points is very simple: First, the user selects the connection by clicking on it. This reveals the tool box with the button for moving control points selected by default, so the user can start moving control points:

- First, he simply places his mouse on a control point (if the pointer is near enough to the control point, it will be highlighted in red).
- Then he selects the control point by clicking on it and keeping the mouse pressed.
- To move the control point he drags the mouse (an overlay showing the new shape of the arc is displayed)
- Finally, he validates the new position of the control point by releasing the mouse.

The state diagram Figure 27 gives an overview of the process of moving control points. The activity diagram displayed in Figure 28 shows in further details how control points are moved according to their type (P0, P1, P2 or P3).

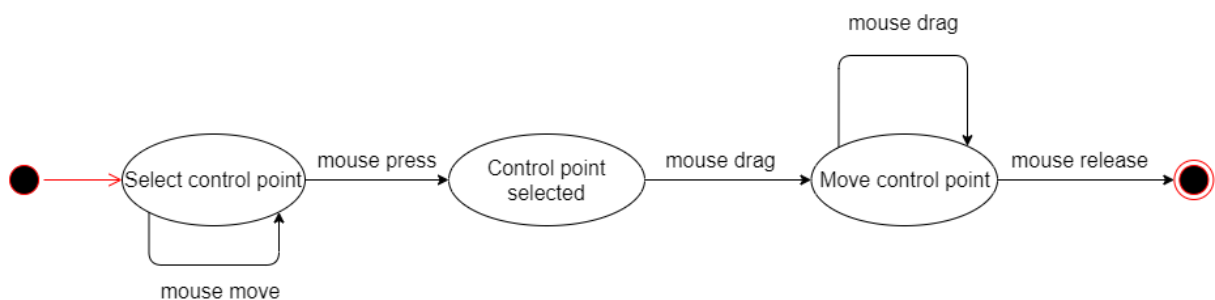


Figure 27: State diagram: moving control points

When the user releases the mouse, the new position of the moved control point is validated and the connection shape stored in the model is updated.

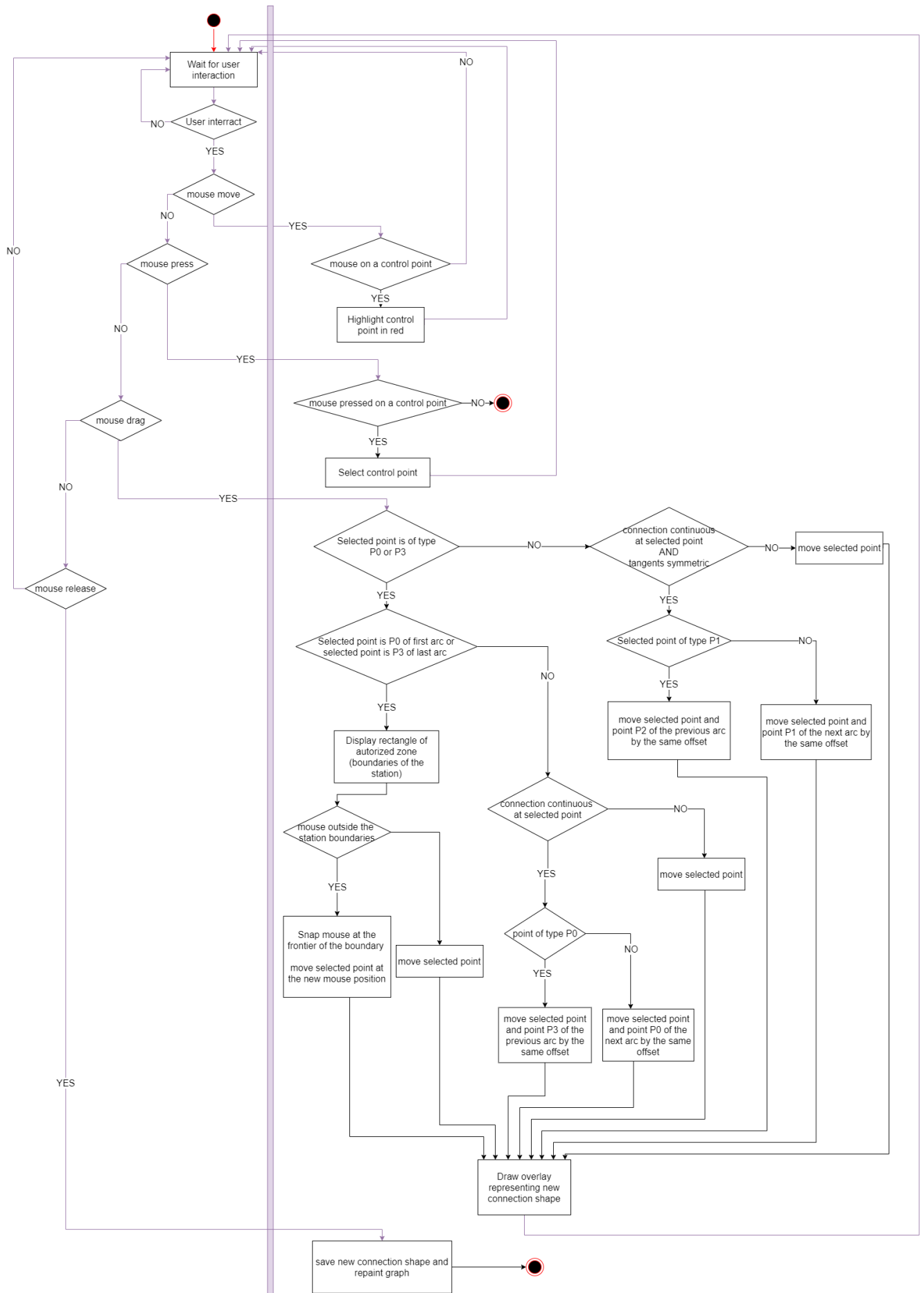


Figure 28: Activity diagram: moving control points

ii. Adding intermediary points

The user also has the possibility to add a new control point on any arc of a connection. The exact position of the new control point cannot be precisely chosen by the user as the added point will be automatically placed in the middle of the selected arc. The position of this point can then be modified as explained in section “Moving controls points”.

Figure 29 shows the addition of a new control point in the last arc. The figure on top shows the selection of the arc (the arc is highlighted in red) and the bottom figure shows the new point.

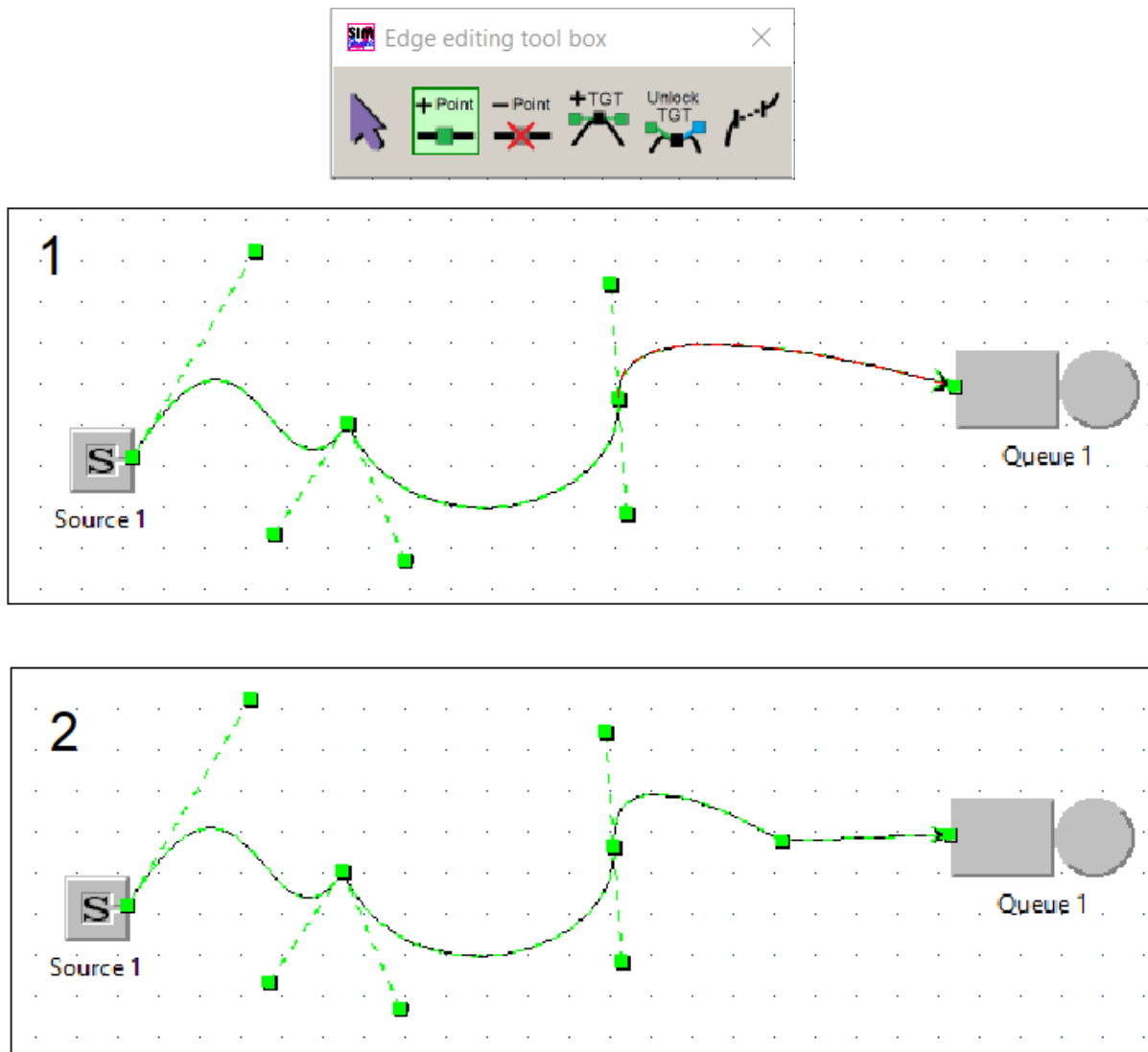


Figure 29: Adding a point on the last arc

In order to add a control point, the user must follow the following steps:

- First, he selects the connection by clicking on it.
- This reveals the tool box on which the user must select the second button.
- Then the user simply places his mouse on the arc on which he wants to add the control point (the arc is highlighted in red if the mouse is close enough to arc).
- Finally, he confirms by clicking on the arc (pressing the mouse and releasing) and a new control point is inserted half way through of the selected arc.

The state diagram Figure 30 gives an overview of the process of adding control points.

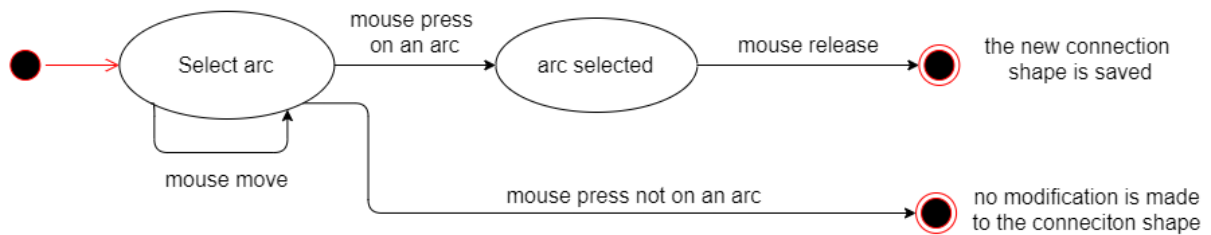


Figure 30: State diagram: adding control point

The activity diagram displayed in Figure 31 shows in further details how a connection shape is modified when a new intermediary point is added.

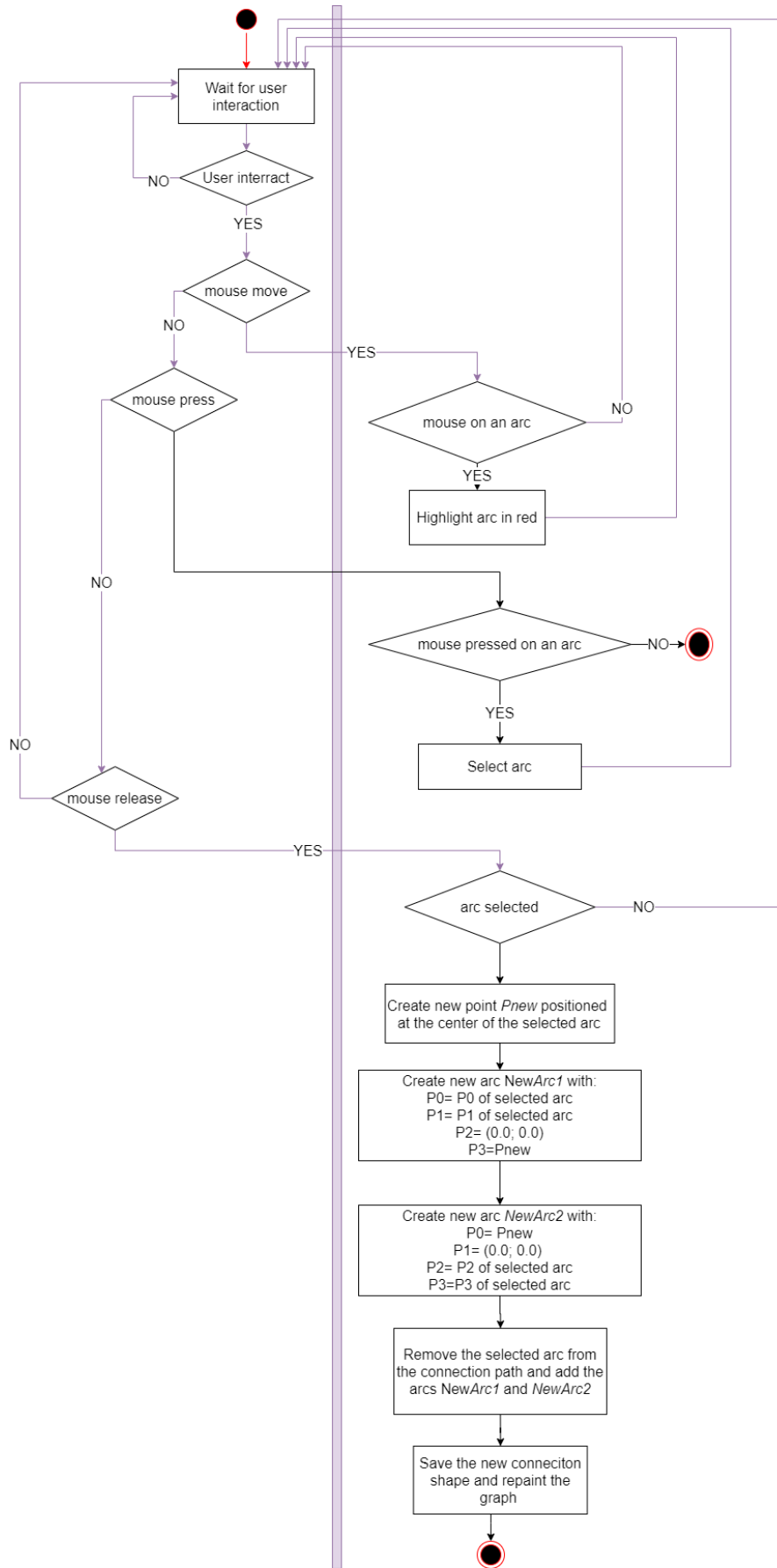


Figure 31: Activity diagram: adding intermediary point

iii. Deleting control points

The user can delete both tangents points (P1 and P2) and intermediary points (P0 and P3) as in Figure 32. When a tangent point is deleted, the tangent becomes null. When an intermediary point is deleted the two arcs respectively originating and leaving that point are replaced by one unique arc.

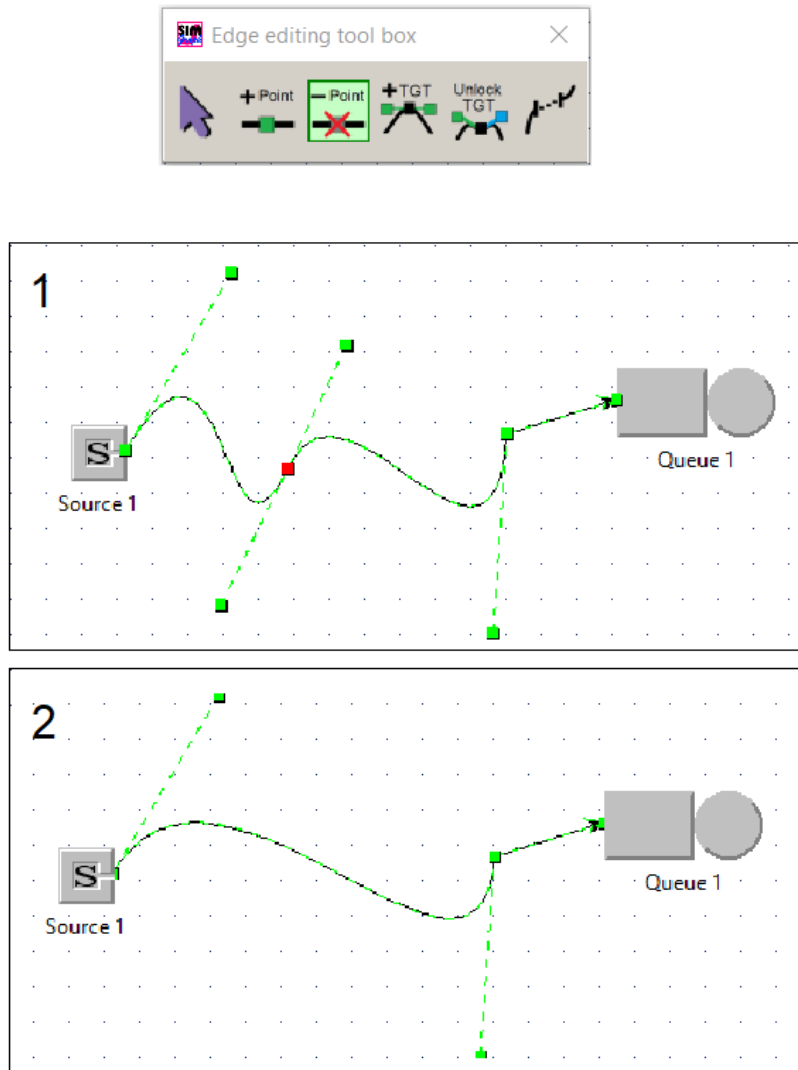


Figure 32: Removing an intermediary point

In order to delete a control point, the user must follow the following steps:

- First, he selects the connection by clicking on it.
- This reveals the tool box on which the user must select the third button.
- Then the user simply places his mouse on the control point that he wants to delete (the point is highlighted in red if the mouse is close enough to it).
- Finally, he confirms by clicking on the control point (pressing the mouse and releasing) and the control point is removed.

The state diagram at Figure 33 gives an overview of the process of adding control points. This state diagram is valid for deleting points, adding tangents and breaking connection. In the case of the point deletion, a “valid point” is any control point except the point P0 of first arc and the point P3 of the last arc.

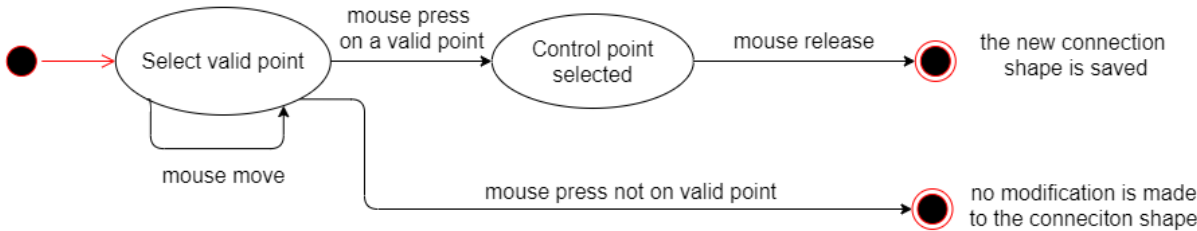


Figure 33: State diagram for deleting points, adding tangents and breaking connection

The activity diagram displayed in Figure 34 shows in further details how a connection shape is modified when a control point is removed.

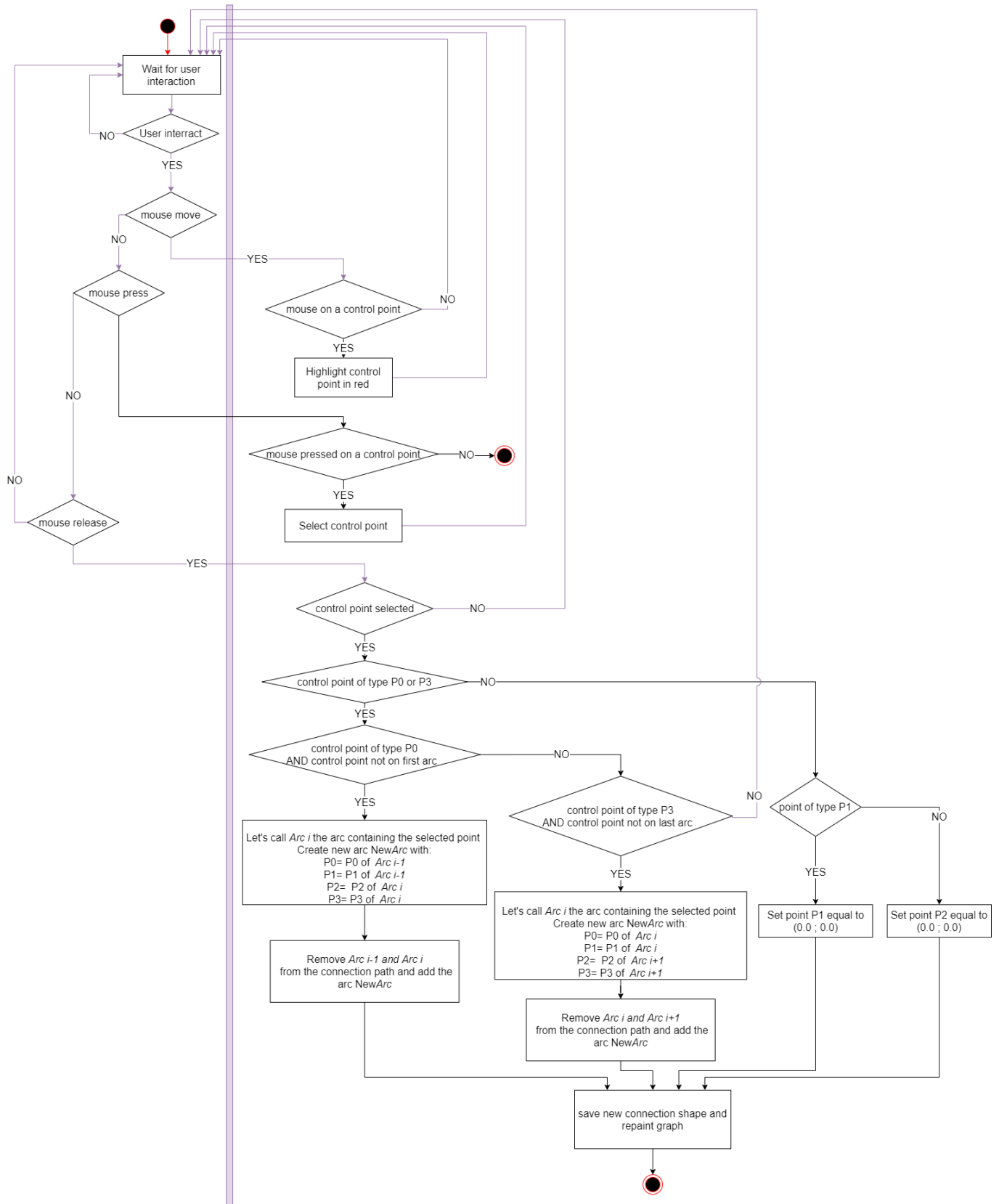


Figure 34: Activity diagram: delete control point

iv. Adding tangents to sharp intermediary points

A sharp intermediary point is a point of type P0 or P3, with a null tangent. As the coordinates of the points P1 and P2 are relative respectively to the points P0 and P3, a null tangent means that, for the considered arc, the point P1 or the point P2 is null. Figure 35 shows symmetrical tangents being added for point P3 of Arc 1 and for point P0 of Arc 2.

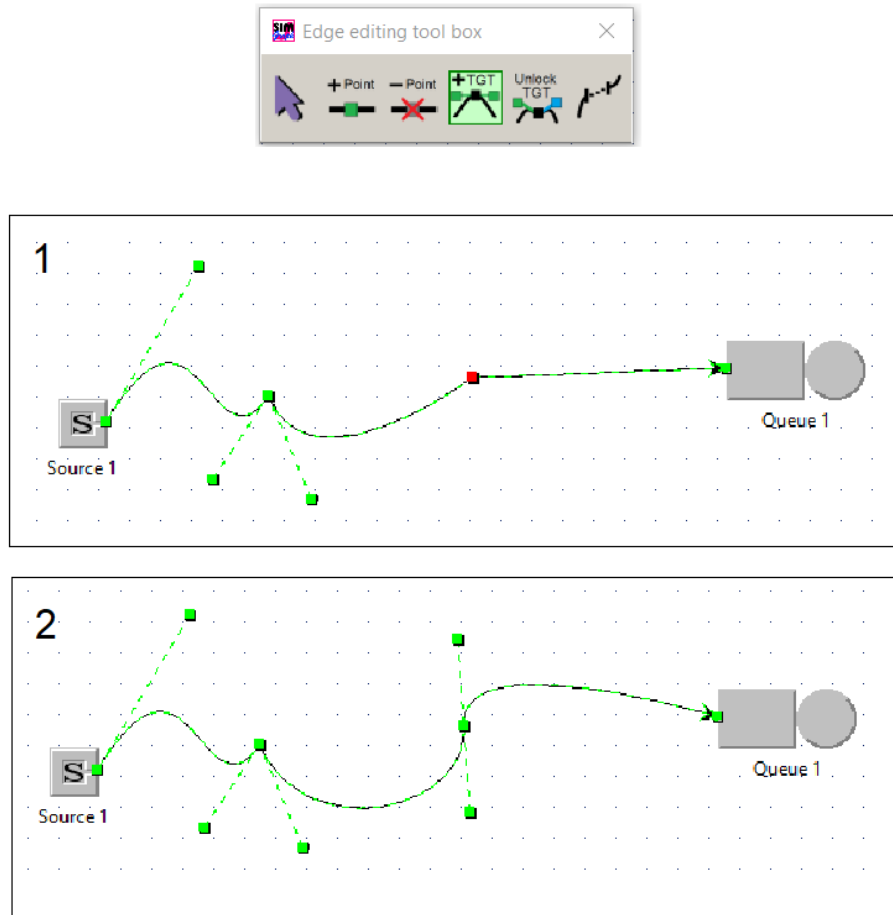


Figure 35: adding tangent to a sharp intermediary point

In order to add tangents to a control point, the user must follow the following steps:

- First, he selects the connection by clicking on it.
- This reveals the tool box on which the user must select the fourth button.
- Then the user simply places his mouse on the control point for which he wants to add tangents (the control point is highlighted in red if the mouse is close enough to it and if it is a sharp control point).
- Finally, he confirms by clicking on the control point (pressing the mouse and releasing) and the tangents are added.

The state diagram Figure 33 gives an overview of the process of adding tangents to a sharp control point. In this situation, a “valid point” is any sharp control point.

The activity diagram displayed in Figure 36 shows in further details how a connection shape is modified when tangents are added to a sharp control point.

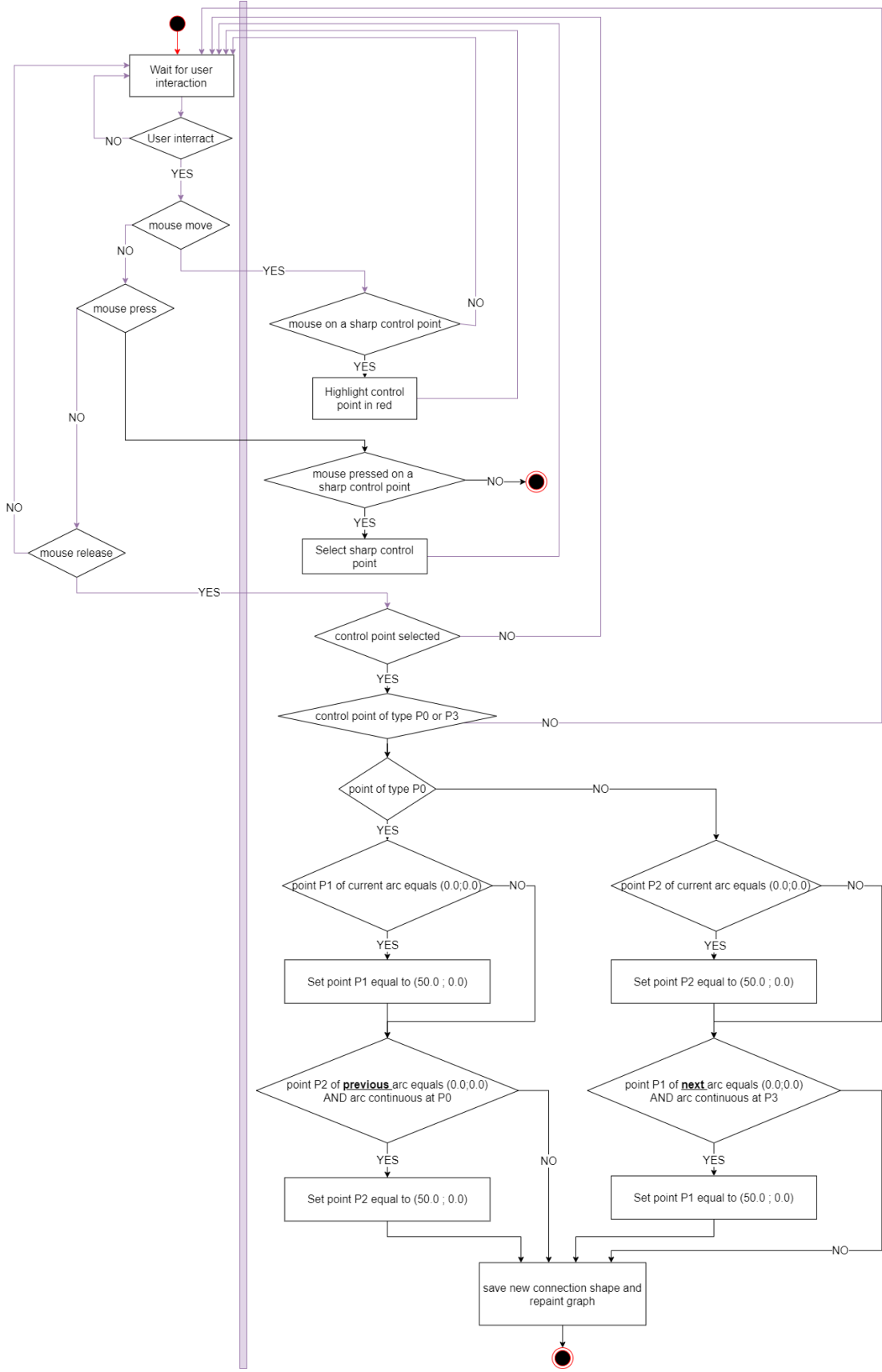


Figure 36: Activity diagram: Add tangents to sharp intermediary point

v. Breaking the tangent symmetry

As seen in Figure 24, the tangents (Points of P1 and P2) are symmetric by default. This means that if a point P1 is moved, and if the connection is continuous at point P0, the point P2 of the previous arc will also be moved in order to keep the symmetry.

If the user wants to break the symmetry, this is possible using the fifth button of the tool bar: the fifth functionality of the tool bar allows the user to move only one side of the tangent.

Figure 37 shows the point P1 of the second arc being moved independently of the other tangent point. Previously the position of this point was linked to the position of the point P3 of the first arc, but the symmetry between those two points has now been broken.

Once the symmetry has been broken, the tangents will be moved independently using the first button of the tool bar.

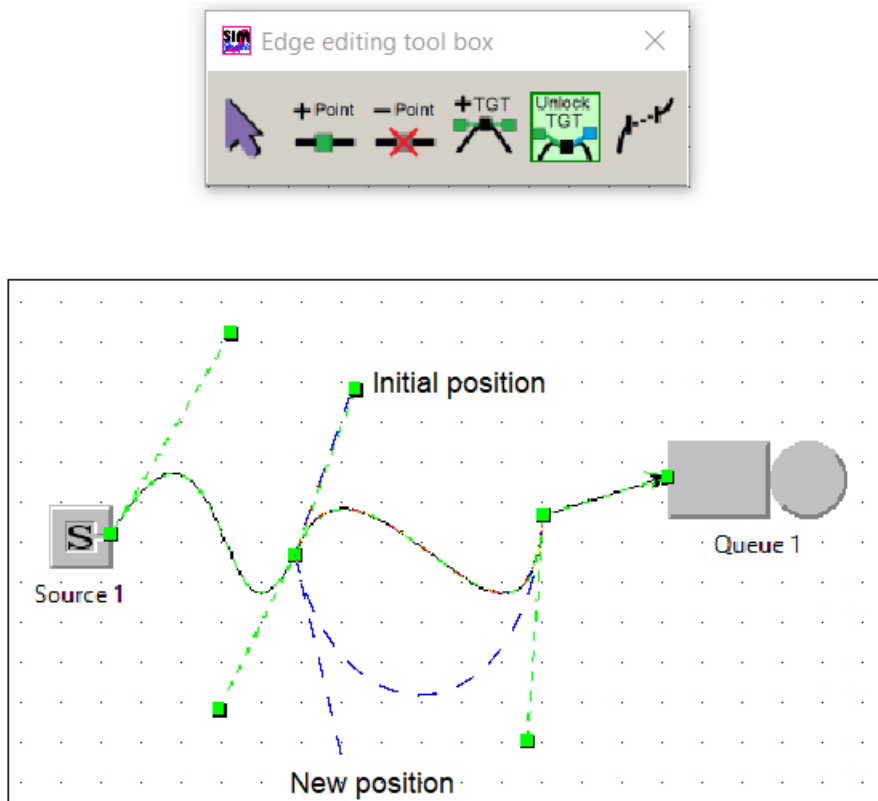


Figure 37: Breaking the tangent symmetry

The process of breaking the tangents symmetry is very simple: First, the user selects the connection by clicking on it. This reveals the tool box on which the user must select the fifth button:

- First, he simply places his mouse on a tangent point (if the pointer is near enough to the control point, it will be highlighted in red).
- Then he selects the tangent point by clicking on it and keeping the mouse pressed.
- To move the tangent point, he drags the mouse (an overlay showing the new shape of the arc is displayed)

Finally, he validates the new position of the tangent point by releasing the mouse.

Figure 38 gives an overview of the process of moving tangents points by breaking the symmetry. The activity diagram displayed in Figure 39 shows in further details how a connection shape is modified when tangent symmetry is broken.

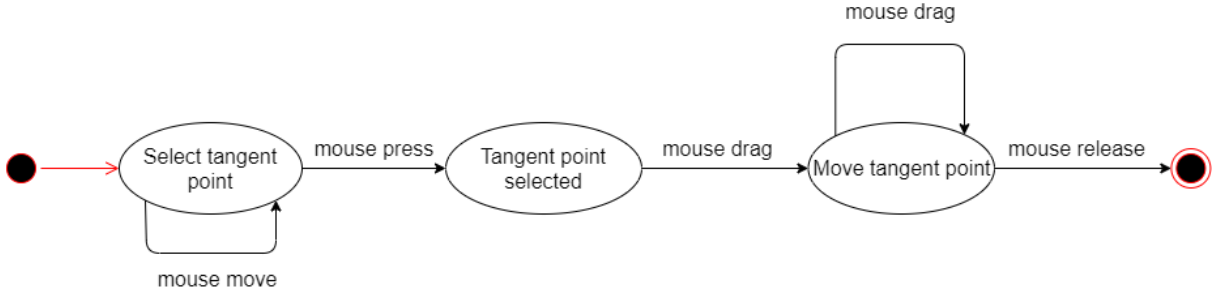


Figure 38: State diagram: Moving tangent point by breaking the symmetry

When the user releases the mouse, the new position of the moved control point is validated and the connection shape stored in the model is updated.

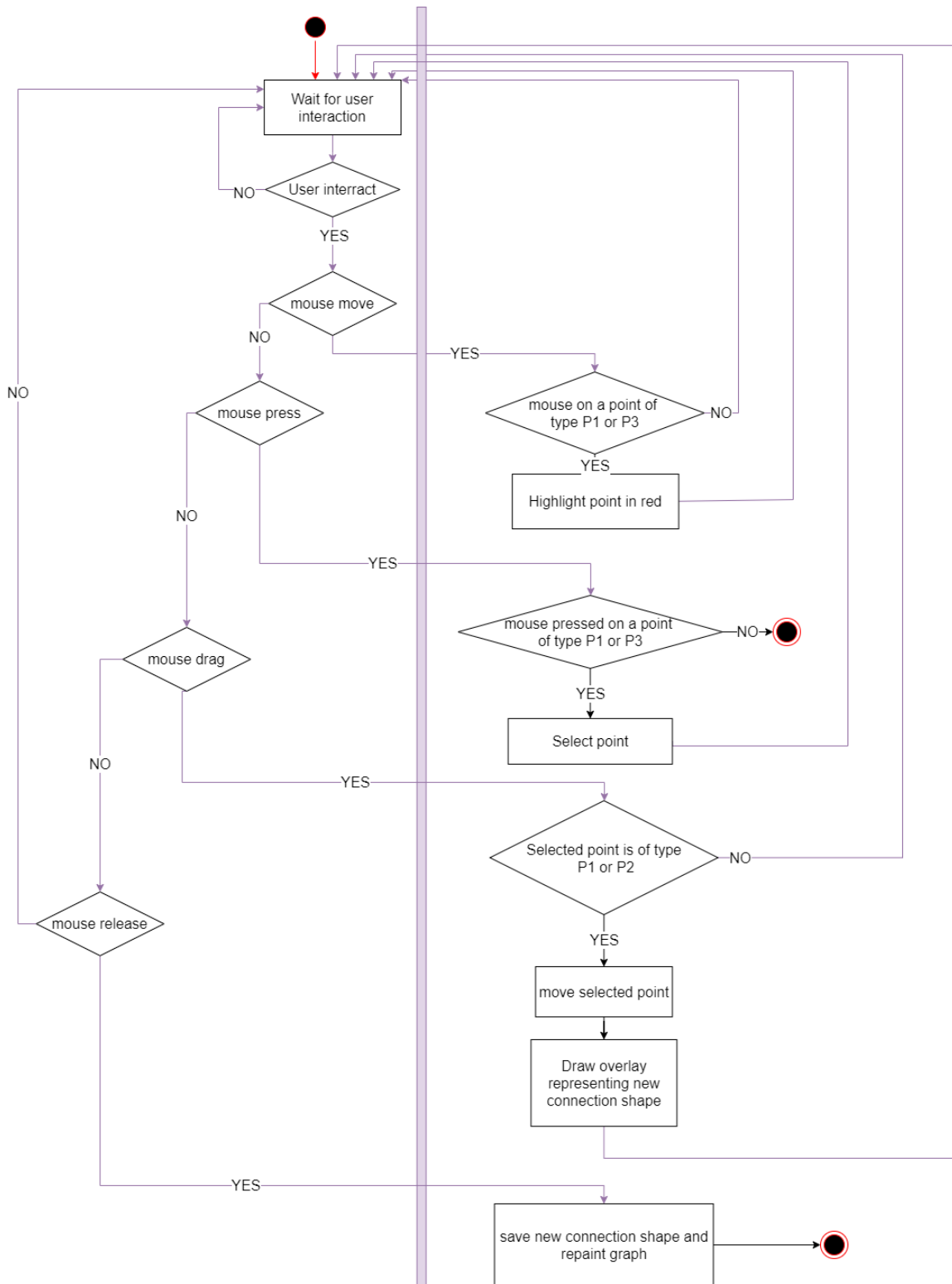


Figure 39:Activity diagram: Breaking the tangent symmetry

vi. Breaking the connection continuity

As seen in Figure 24, the connections are continuous by default. This means that for a connection containing n arcs, $\forall i \in [0, n - 2]$, the point P3 of arc i and the point P0 of arc $i + 1$ coincide. This means that if a point P3 is moved by a given offset, and if the connection is continuous at point P3, the point P0 of the next arc will also be moved by the same offset in order to keep the continuity.

If the user wants to break the continuity, this is possible using the sixth button of the tool bar: the sixth functionality of the tool bar allows the user to separate the point P3 of arc i and the point P0 of arc $i + 1 \forall i \in [0, n - 2]$ with n the number of arcs in the connection.

Figure 40 shows the breakage of the continuity between the penultimate arc and the last arc. The figure on top shows the selection of the breaking point (the intermediary point is highlighted in red) and the bottom figure shows the result.

Once the continuity has been broken, the intermediary points P0 and P3 will be moved independently using the first button of the tool bar. This allow the user to merge the arcs back together by moving the broken pieces so that they coincide. Breaking the continuity also imply breaking the symmetry so the tangents will also be moved independently until the arcs are merged back together.

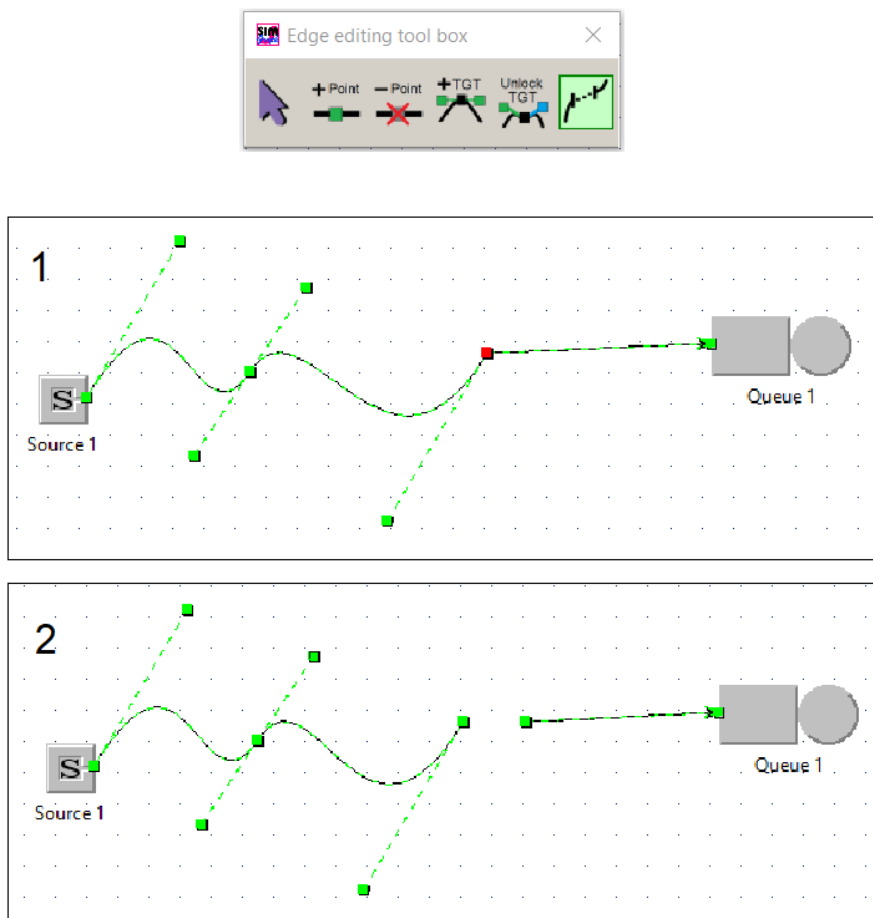


Figure 40: Breaking the arc continuity

In order break a connection, the user must follow the following steps:

- First, he selects the connection by clicking on it.
- This reveals the tool box on which the user must select the sixth button.
- Then the user simply places his mouse on the intermediary point where he wants to break the arc (the control point is highlighted in red if the mouse is close enough to it).
- Finally, he confirms by clicking on the control point (pressing the mouse and releasing) and the connection is broken.

A connection can be broken at any intermediary point except at the source station (point P0 of first arc) or at the target station (point P3 of the last arc).

The state diagram Figure 33 gives an overview of the process of breaking a connection from the user's perspective. In this situation, a "valid point" is any intermediary point except point P0 of first arc or point P3 of the last arc.

The activity diagram displayed in Figure 41 shows in further details how a connection shape is modified when the continuity is broken.

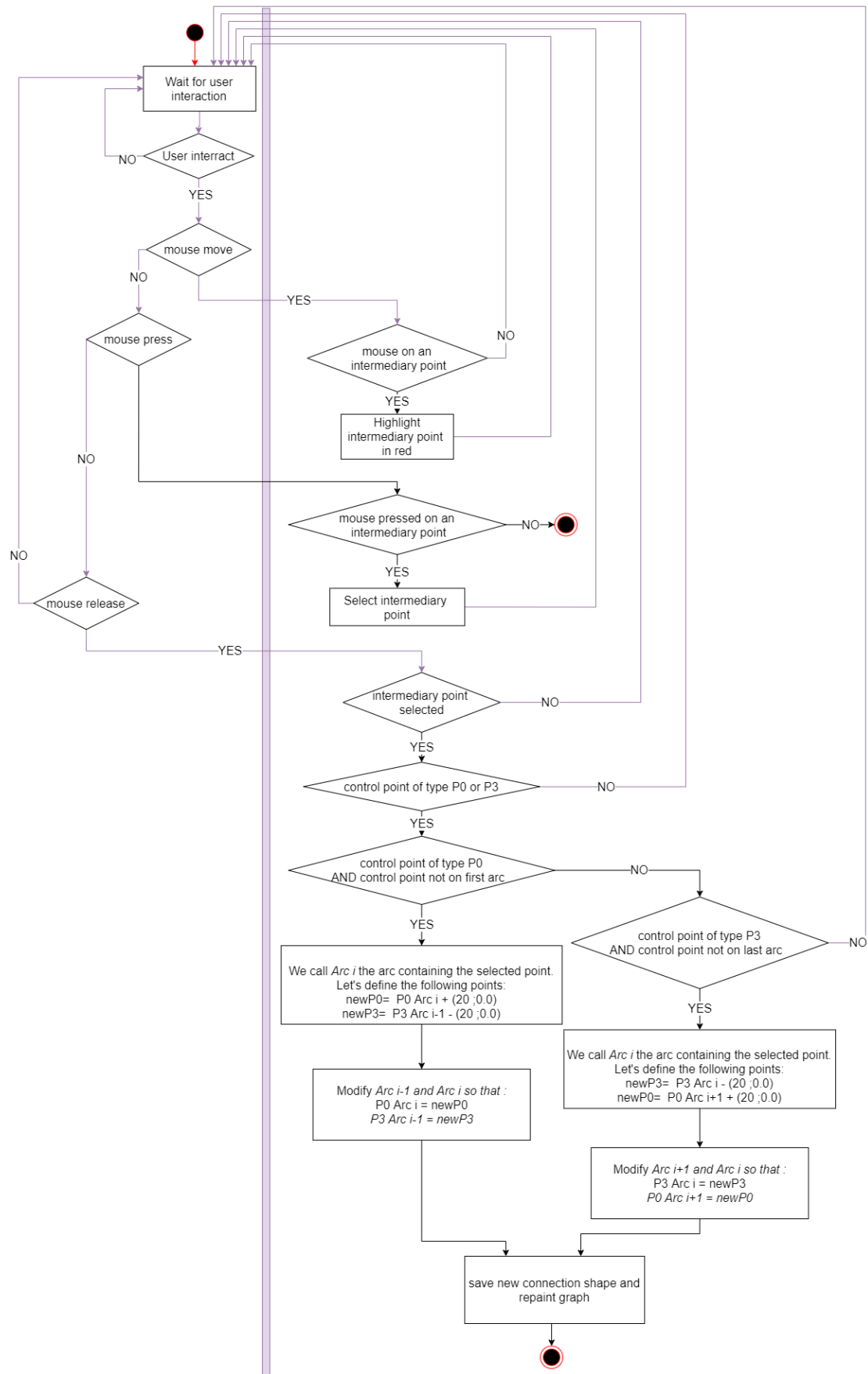


Figure 41: activity diagram: Breaking the connection continuity

vii. [Possibility to switch between connection types.](#)

As explained before, the user can choose between connecting stations with an automatic and non-editable connection or with a Bezier connection that has to be hand drawn and that can be modified afterward.

It is also possible to change the type of a connection already created. This way it is possible to update the connections of a network created with an older version of JMT (where the only available connection type was the automatic one).

The connection type can be modified in the two ways: automatic to Bezier and Bezier to automatic. The modification of a connection type only impacts the appearance of a connection: it doesn't influence the input parameters of the model like the firing rules.

4. Station rotation

As described in section "The impossibility to rotate stations", JMT did not give to the user the possibility to rotate stations. As this could compromise the clarity of networks layouts, functionalities allowing the rotation of stations by hops of 45° (to the left or to the right) were added to the tool.

The rotation of stations is made possible only for stations that are only connected with Bezier connections. This choice was made because the algorithm in charge of automatically computing the shape of non-Bezier connection is not made to take in consideration the fact that stations can be rotated.

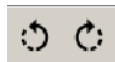
In order to prevent the rotation of stations connected by a non-Bezier connection, two new attributes were added to the definition of a station (class `JmtCell`)

- *isFreeRotationAllowed*: a Boolean equal to *true* if the station is linked only to Bezier connections and *false* otherwise.
- *rotationAngle*: a double representing the rotation angle in radians. The attribute *rotationAngle* is always equal to 0.0 if *isFreeRotationAllowed* is *false*.

Each time a new connection is added, or each time a connection type is changed, a test is performed to check if the status of the attribute *isFreeRotationAllowed* should be changed for the source or target station.

In order to rotate a station the user must:

1. Select the station
2. Click on one of the two buttons for rotating stations:



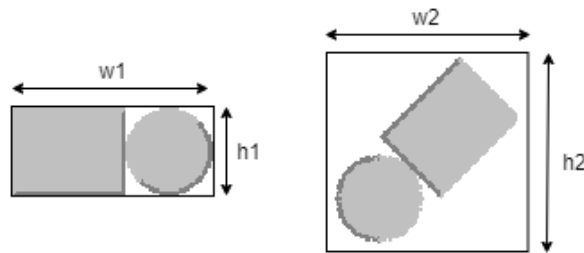
If the free rotation is not authorized for the selected station, the user will not be able to perform the step 2 because the rotations buttons will be disabled.

It is possible to combine rotation and mirroring of station without restrictions.

The rotation of a station implies the rotation of the station icon, the rotation of the ports and the rotation of the end of the Bezier connections linked to that station.

i. The image rotation

When a station is rotated, the image of the station must be reloaded. The image of the station is rotated by the specified rotation angle using an affine transformation, and the size of the image is changed accordingly.



If we call θ the rotation angle in radians, the new width (w_2) and the new height (h_2) are determined using the standard image width (w_1) and height (h_1) according to the following formula:

$$w_2 = w_1 * \cos(\theta) + h_1 * \sin(\theta)$$

$$h_2 = w_1 * \sin(\theta) + h_1 * \cos(\theta)$$

ii. The ports rotation

The input and output ports position are defined as an offset from the top left edge of the station bounds. The bounds of the station are delimited by a rectangle encapsulating the station icon and the station name (the dotted green rectangle in Figure 42). When a station is being rotated or mirrored, the ports positions need to be updated.

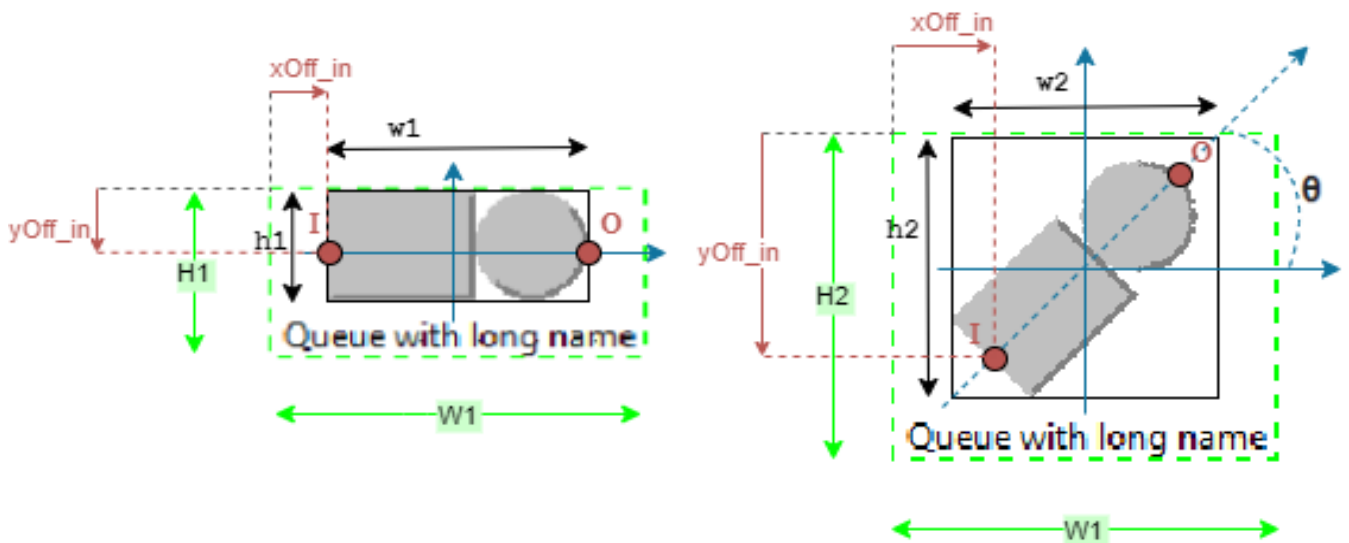


Figure 42: illustration of ports rotation

Legend:	
h1 :	height of the standard station icon (rotation angle of 0°)
w1 :	width of the standard station icon (rotation angle of 0°)
H1:	height of the station bounds before rotation
W1:	width of the station bounds before rotation
h2 :	height of the station icon after rotation
w2:	width of the standard station icon after rotation
H2:	height of the station bounds after rotation
W2:	width of the station bounds after rotation
O:	output port
I:	input port
Θ:	rotation angle
xOff_in:	x Offset of the input port
yOff_in:	y Offset of the input port

Figure 42 shows a queue station before and after the rotation. In order not to unnecessarily overload the figure, only the offset for the input port are presented. The offsets for the output port are defined, as for the input port, from the top left corner of the rectangle defining the station bounds.

The offset coordinates for both the input and output port are:

$$xOff = \frac{W2 - w2}{2} + x \cdot \frac{1000}{W2}$$

$$yOff = \frac{y}{H2} * 1000$$

The values of x and y depends of the nature of the port:

- For the input port of a non-mirrored station, or for the output port of a mirrored station:

$$x = \frac{w1}{2} * \cos(\theta + \pi) + \frac{w2}{2}$$

$$y = \frac{w1}{2} * \sin(\theta + \pi) + \frac{h2}{2}$$

- For the input port of a mirrored station, or for the output port of a non-mirrored station:

$$x = \frac{w1}{2} * \cos(\theta) + \frac{w2}{2}$$

$$y = \frac{w1}{2} * \sin(\theta) + \frac{h2}{2}$$

iii. The rotation of connections shapes

For any connection, the user has the possibility the move point P0 of the first arc and the point P3 of the last arc anywhere inside the boundary of the station. This means that the connections do not necessary originate from the output port of a station, and do not necessary arrives at the input port of a station.

- If a connection originates from the output port of a station, the point P0 of the first arc then, $P0_{firstArc} = (0.0 ; 0.0)$.
- If a connection goes to the input port of a station, the point P3 of the last arc then, $P3_{LastArc} = (0.0 ; 0.0)$.

If the user moves the last point of a connection (in order to avoid overlapping for example) then the point $P3_{LastArc}$ in the JMTPath will be modified. After being moved, the value of $P3_{LastArc}$ is equal to an offset from the input port position.

The behavior is the same if the user moves the first point of a connection but in this case the value of $P0_{firstArc}$ is equal to an offset from the output port position.

Figure 43 gives an example of a queue station receiving three connections in input. In this situation the point P3 of the last arc for the connection at the top and for the connection at the bottom have been moved: they don't coincide with the input port

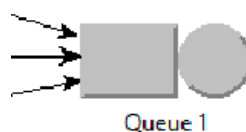


Figure 43: Queue station with three input connections

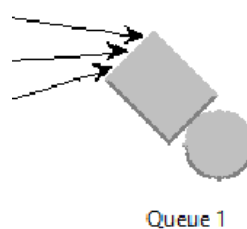


Figure 44: Queue station with three input connections after rotation

When the station is rotated as in Figure 44, the values of $P3_{LastArc}$ for the top and bottom connection have been updated to take the rotation into account.

The Figure 45 shows a queue station before and after a rotation by an angle θ . For the connection on top, the initial value of $P3_{LastArc}$ is given by the vector $\vec{v1}$, and the value after rotation is given by the vector $\vec{v2}$.

The vector \vec{i} is the unit vector $\vec{i} = [1 \ 0]$.

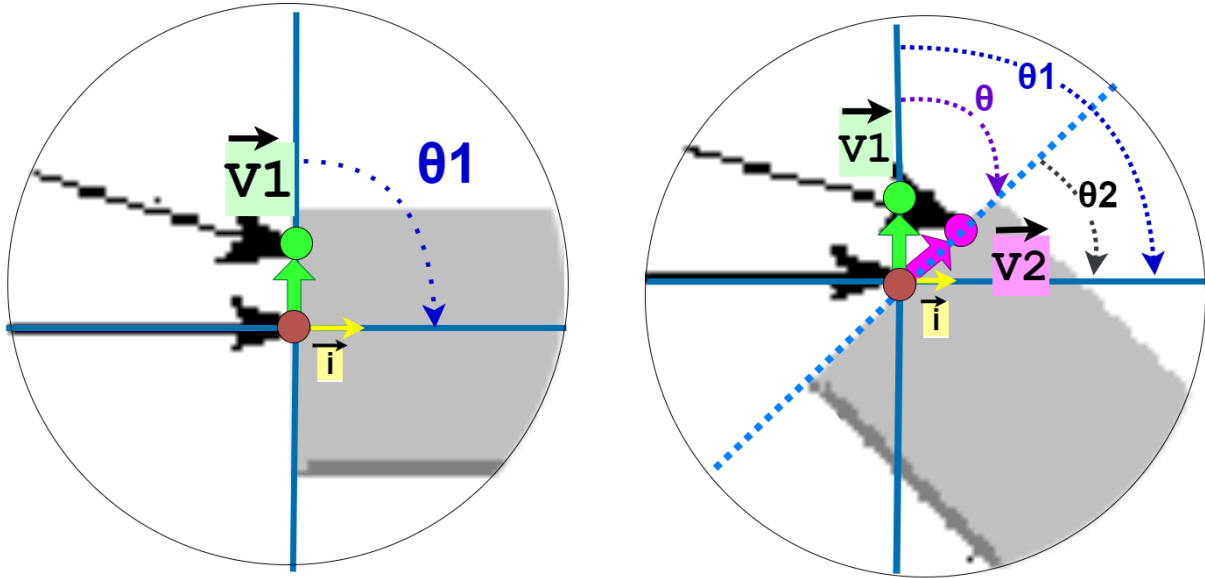


Figure 45: Illustration for the calculation of connection rotation

The coordinates of the vector $\vec{v2}$ are obtained by the following method:

$$\theta1 = \arccos\left(\frac{\vec{v1} \cdot \vec{i}}{\|\vec{v1}\| \cdot \|\vec{i}\|}\right)$$

$$\theta2 = \theta1 - \theta$$

$$\vec{v2} = \|\vec{v1}\| * [\cos(\theta2) \quad \sin(\theta2)]$$


Finally, the new value of the point $P3_{LastArc}$ is obtained:

$$New P3_{LastArc} = (\|\vec{v1}\| * \cos(\theta2); \|\vec{v1}\| * \sin(\theta2))$$

This method applies also in the case where the point to be updated is $P0_{firstArc}$.

5. Grid snapping

As explained in section “The difficulty to properly and quickly align stations”, JSIMGraph did not have a functionality allowing users to snap stations or controls points of a connection on a grid.

In order to make easier the process of aligning stations and connections, a grid functionality has been added to JSIMGraph. Now it is possible for the user to activate a grid by pressing the grid button: 

The button activate the grid and displays it to the user (as shown in Figure 46 and Figure 47)

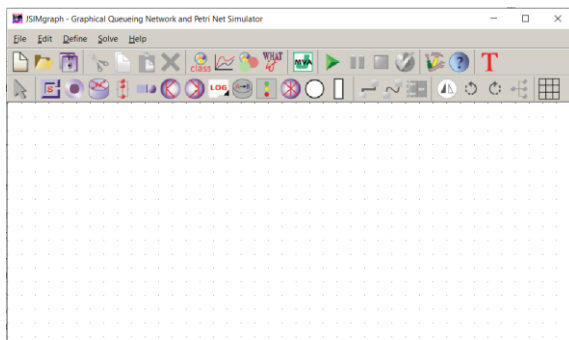


Figure 46: GUI with grid disabled

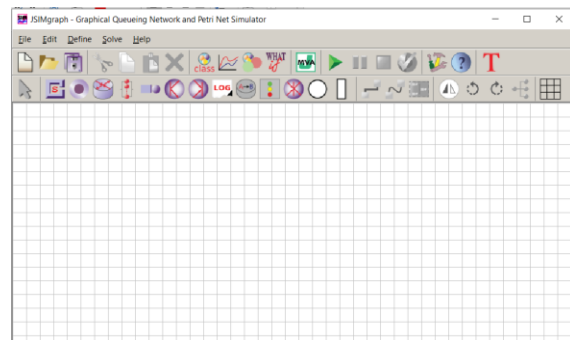


Figure 47: GUI with grid enabled

When the grid is enabled, the stations will be snapped at the grid by their output port. This means that the port of the stations will be placed at the intersection between two grid lines.

When the grid is active, the snapping occurs each time the user inserts a station, moves a station, draws a Bezier edge or moves the control points of a Bezier edge.

The snapping is made using the already available function *snap* of the Swing based software component JGraph.

VI. Other extensions ideas

Other features could further improve the tool JSIMGraph. The improvements ideas presented in this section were not originally planned, they emerged during the development of the already presented features.

i. Add identifier to broken connections

As explained in section “Breaking the connection continuity”, a new feature has been developed allowing the users to break connections. One issue is that when several connections are broken, the only way for the user to know how the broken pieces are connected is to select the connection as in

Figure 48. A solution would be to add an alphabetical identifier to the broken connection as shown in Figure 49.

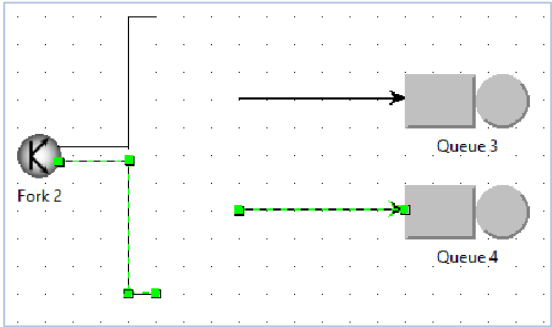


Figure 48: Selected broken arc

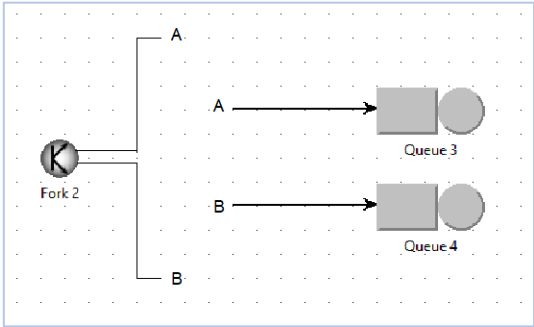


Figure 49: Broken arc with alphabetical identifier

ii. Possibility of deleting a connection and creating a new one by dragging the connection arcs.

Another improvement idea would be to allow users to modify the connections between stations by dragging the connection shapes.

For example in Figure 50, we could allow the user to drag the last control point of the connection between the stations Fork 1 and Queue 1 to the station Queue 2; and by doing so, the connection between Fork 1 and Queue 1 would be deleted and a connection between Fork 1 and Queue 2 would be created.

This action should in addition preserve the Firing parameter provided in station Fork1.

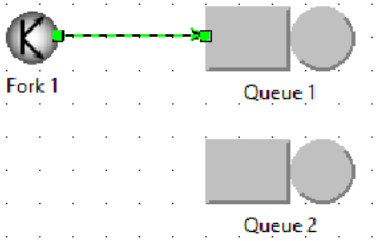


Figure 50: changing connections

VII. Conclusion

This document describes the extensions made to the graphical interface of JMT, a tool used to perform performance evaluation, capacity planning and modelling of computer and communication systems. The four main extensions are the possibility for the user to draw the connection shapes, the possibility to modify the connection shapes, the possibility to rotate stations, and the possibility to activate a grid. In the section “The automatic connection shapes generating a confusing layout”, an example of computer network was presented. This example is repeated below in Figure 51. Figure 52 presents the same network re-made taking advantage of the extension of the graphical interface. In the second version, all the arcs are of Bezier type, and all stations are perfectly aligned using the grid snapping feature. The extremities of the arcs (input and output) are placed in order to avoid arc intersection and overlapping. The comparison between Figure 51 and Figure 52 illustrate that the added features greatly improve the readability of network layout.

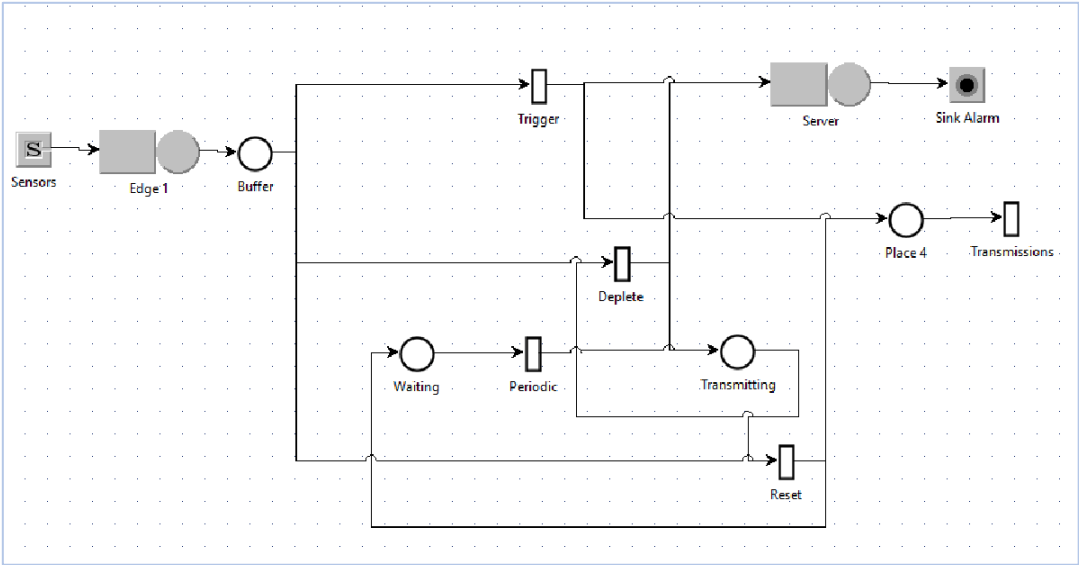


Figure 51: Example of computer network before GUI extension

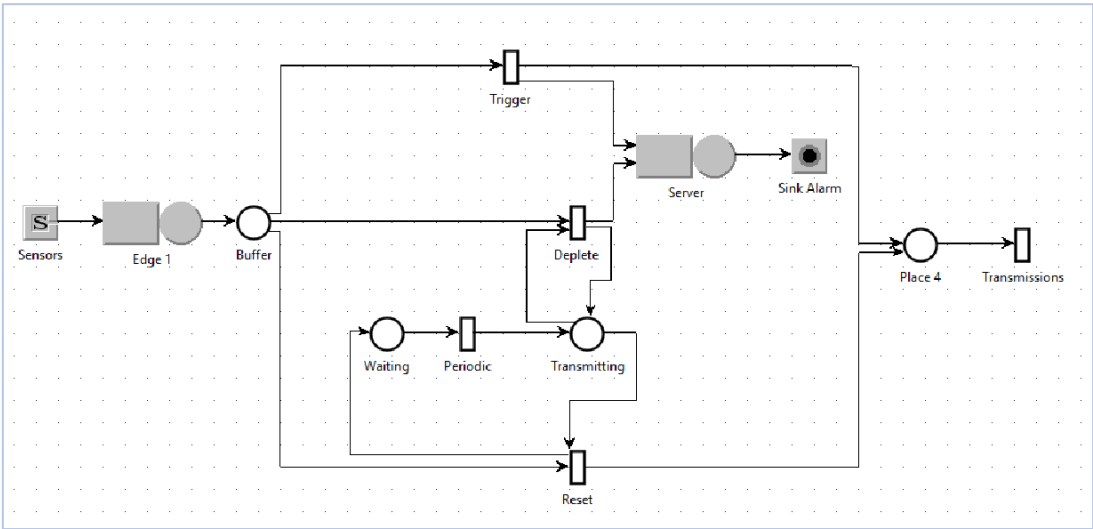


Figure 52: Example of computer network after GUI extension

Other extensions could be added to the GUI in order to further improve it, like allowing the transfer of connections between stations or adding identifiers to the broken arcs (see section “Other extensions ideas”). Indeed, the software JMT is in constant evolution.

The extensions presented in in document have not been released yet, as they still need to be approved by the others stakeholders of the JMT project, but I hope they will soon be available to be downloaded by all users of JMT.

List of tables

Table 1: Values for Service time and visits, first study case	14
Table 2: Stations parameters, first study case	14
Table 3: Simulation results, first study case	15
Table 4: Stations parameters, second study case	17
Table 5: Results on throughput (all classes), second study case.....	18
Table 6: Results on system response time for Class0, second study case	19
Table 7: Results on system response time class1, second study case	20
Table 8: non-functional requirements	24
Table 9: functional requirements.....	25

List of figures

Figure 1: GUI of GreatSPN	3
Figure 2: GUI of Möbius.....	4
Figure 3: GUI PEPA Eclipse Plug-in	5
Figure 4: GUI PIPE2	6
Figure 5: PRISM GUI	7
Figure 6: GUI of SHARPE.....	8
Figure 7: GUI of TANGRAM II.....	8
Figure 8: Start screen of JMT.....	10
Figure 9: Network layout, first study case.....	14
Figure 10: Network layout after simulation, first study case.....	15
Figure 11: Network layout, second study case.....	16
Figure 12: Results on throughput (all classes), second study case	18
Figure 13: Results on system response time for Class0, second study case	19
Figure 14: Results on system response time class1, second study case	19
Figure 15: Model with problematic arc shapes.....	21
Figure 16: highlight of port input and output overlap problem.....	22
Figure 17: Example of station being mirrored.....	22
Figure 18: Example of problem caused by the impossibility to rotate stations.....	23
Figure 19: Example of a nicer layout that could be obtained by rotating stations.....	23
Figure 20: Example of new connection shape.....	26
Figure 21: Example of cubic Bezier curve.....	27
Figure 22: Example of structure of new connection shape.....	27
Figure 23: State diagram of the process of drawing Bezier connections.....	31
Figure 24: Example of a Bezier connection showing the continuity and symmetry of tangents.....	31
Figure 25: moving point P2 of the first arc.....	32
Figure 26: Moving point P0 of the first arc.....	33
Figure 27: State diagram: moving control points.....	33
Figure 28: Activity diagram: moving control points	34
Figure 29: Adding a point on the last arc	35
Figure 30: State diagram: adding control point	36
Figure 31: Activity diagram: adding intermediary point	37
Figure 32: Removing an intermediary point.....	38
Figure 33: State diagram for deleting points, adding tangents and breaking connection.....	39
Figure 34: Activity diagram: delete control point	40

Figure 35: adding tangent to a sharp intermediary point 41

Figure 36: Activity diagram: Add tangents to sharp intermediary point 42

Figure 37: Breaking the tangent symmetry..... 43

Figure 38: State diagram: Moving tangent point by breaking the symmetry..... 44

Figure 39:Activity diagram: Breaking the tangent symmetry 45

Figure 40: Breaking the arc continuity 46

Figure 41: activity diagram: Breaking the connection continuity 48

Figure 42: illustration of ports rotation..... 50

Figure 43: Queue station with three input connections 52

Figure 44: Queue station with three input connections after rotation 52

Figure 45: Illustration for the calculation of connection rotation..... 53

Figure 46: GUI with grid disabled 54

Figure 47: GUI with grid enabled..... 54

Figure 48: Selected broken arc..... 55

Figure 49: Broken arc with alphabetical identifier 55

Figure 50: changing connections..... 55

Figure 51: Example of computer network before GUI extension 56

Figure 52: Example of computer network after GUI extension 56

Bibliography

Anonymous, (2009), <i>The PEPA Plug-in Project</i> , http://www.dcs.ed.ac.uk/pepa/tools/plugin/index.html	5
Anonymous, <i>PIPE2 - Platform Independent Petri net Editor 2</i> http://pipe2.sourceforge.net/index.html	5
BABAR J., BECCUTI M., DONATELLI S., MINER A., (2010), <i>GreatSPN Enhanced with Decision Diagram Data Structures</i>	3
BONET P., LLADO C.M., PUIJANER R., KNOTTENBELT W.J., (2007), <i>PIPE v2.5: A Petri Net Tool for Performance Modelling</i>	5
CASALE G., MUNTZ R. R., SERAZZI G., (2009), <i>Special Issue on Tools for Computer Performance Modeling and Reliability Analysis</i>	3
CASALE G., SERAZZI G., (2011), <i>Quantitative System Evaluation with Java Modeling Tools (Tutorial Paper)</i>	9
CASALE G., SERAZZI G., BERTOLI M., (2020), <i>Java Modelling Tools- User manual</i>	9
COURTNEY T., DALY D., DERISAVI S., GAONKAR S., GRIFFITH M., LAM V., SANDERS H. W., (2004), <i>The Möbius Modeling Environment: Recent Developments</i>	4
DE-SOUZA-E-SILVA E., FIGUEIREDO D. R., LEO R. M.M., (2009), <i>The TANGRAM-II Integrated Modeling Environment for Computer Systems and Networks</i>	8
Duke University SHARPE Portal, (2020), <i>SHARPE Portal</i> , https://sharpe.pratt.duke.edu/	7
KWIATKOWSKA M., NORMAN G., PARKER D., (2011), <i>PRISM 4.0: Verification of Probabilistic Real-time Systems</i>	6
Laboratory for Foundations of Computer Science, The University of Edinburgh, (2011), <i>PEPA – Performance Evaluation Process Algebra</i> , http://www.dcs.ed.ac.uk/pepa/	4
LAND (Laboratory for modeling, analysis and development of networks and computer systems), <i>Tangram – II</i> , http://www.land.ufrj.br/tools/tools.html	8
LAZOWSKA E. D., ZAHORIAN J., GRAHAM G. S., SEVCIK K. C. (1984). <i>Quantitative System Performance - Computer System Analysis Using Queueing Network Models</i> . Prentice-Hall	2
Möbius, (2020), <i>Möbius – Model-based environment for validation of system reliability, availability, security, and performance</i> , https://www.mobius.illinois.edu/research.php	4
PINCIROLI R., GRIBAUDO M., ROVERI M., SERAZZI G. (2018) <i>Capacity Planning of Fog Computing Infrastructures for Smart Monitoring</i>	21
PRISM, (2020), <i>Prism Model Checker</i> , http://prismmodelchecker.org/	6
SMITH M J. A., (2010), <i>A Tool for Abstraction and Model Checking of PEPA Models</i>	5
TRIVEDI K. S., SAHNER R., (2009), <i>SHARPE at the age of twenty-two</i> , SIGMETRICS Perform	7