FREE AND OPEN-SOURCE SIMULATION SOFTWARE "URURAU"

Túlio Almeida Peixoto
Universidade Candido Mendes (UCAM-Campos)
100 Anita Pessanha St, Pq. São Caetano.Campos dos Goytacazes, RJ, 28030-335, BRAZIL
tulioap@gmail.com

João José de Assis Rangel
Universidade Candido Mendes (UCAM-Campos)
joao@ucam-campos.br

Ítalo de Oliveira Matias
Universidade Candido Mendes (UCAM-Campos)
italo@ucam-campos.br

RESUMO

O Ururau é um software livre e de código fonte aberto escrito na linguagem de programação Java. A utilização do software permite a construção de modelos de simulação discreta nas três camadas que compõem a estrutura do software. Isto quer dizer que os modelos podem ser construídos na camada superior da interface gráfica de forma mais rápida e de simples programação, no núcleo do software (novos comandos, por exemplo) ou na camada inferior das bibliotecas de software. A aplicação do Ururau tem possibilitado a realização de pesquisa de novas funções e algoritmos no campos da simulação de sistemas a eventos discretos.

PALAVARAS CHAVE. Simulação a eventos discretos, Software livre, Ururau.

Área principal: SIM – Simulação.

ABSTRACT

Ururau is a free and open-source software written in the Java programming language. The use of the software allows the construction of discrete simulation models on the three layers that constitute the structure of the software. It means that the models can be developed in the upper layer of the graphic interface faster and of simple programming in the core of the software (new commands for example) or in the lower layer of the software libraries. The use of the Ururau has made the accomplishment of research of new functions and algorithms in the field of discrete event systems simulation possible.

KEYWORDS. Discrete event simulation. Free and open-source software. Ururau.

Main area: SIM – Simulation.

1. Introduction

This paper aims to present a software, named Ururau, which can be used in the construction of Discrete Event Simulation (DES) models. The Ururau is a Free and Open-Source Software (FOSS) developed in Java by a group of researchers interested in understanding and promoting the design and structure of the source code of a software for development of DES models. It is possible to develop simulation models using Ururau both in a GUI of simple use as in Java, in the lowest layer of software, or even interconnect modules of the JSL simulation library in the middle layer. The origin of the word Ururau refers to a legend of the eighteenth century about an alligator that lived in the river that goes through the city where the researchers of this project live (Peixoto; Rangel, and Matias, 2014). We emphasize that, similarly to Ururau software, there is also the JaamSim.

The JaamSim software, according to King and Harrison (2013), was the first open-source simulation software to include a modern GUI that is comparable to commercial off-the-shelf simulation software. We do not know for sure whether or not the Ururau preceded the JaamSim. However, the main thing is to know that there are people interested in building FOSS with the idea of creating DES models. Within this perspective, it is important to mention the work of Swain (2007), which presents a descriptive and detailed list on more than fifty commercial software that can be used in the development of DES models. We do not have the intention to raise any questions about advantages or disadvantages of commercial or free software, just show that there is still much to do regarding FOSS.

Also according to King and Harrison (2013), previous simulation engines of Open code in Java include CSIM, DESMO-J, GOD, DSOL, JavaSim, JIST, Jsim, JSL, SimJava, Simkit, SSJ, and Tortuga. They cite that although these packages provide many useful features that make them good choices for some applications, each one only provides a library of simulation tools. In this case, the user is responsible for preparing the model from Java code without the existence of a Graphical User Interface (GUI). In such cases, the modeler must have the ability to program the models in this way.

On the other hand, in a more educational context, there is the proposal of Lawson and Leemis (2009). The authors suggest the use of Simulation 101. This software has, basically, random number generators and a mechanism to generate events, being all the control of the simulation practically done by the analyst who programs the model. However, the Simulation 101 proves to be quite useful as an introduction to the topic of simulation for students of Computer Science and Industrial Engineering.

In the case of Ururau, the simulation library Java Simulation Library (JSL), proposed by Rossetti (2008), is applied. The JSL allows the construction of models oriented to process. When necessary, it also permits the addition of new process commands when the simulation model becomes more complex. Thus, by employing the Java language, the Ururau is multiplatform and has GPL license (General Public License) for development in open source.

Finally, it is emphasized that the Ururau has been built as a kind of research laboratory in DES. We offer the Ururau as a true open-source GPL project for simulation community. There are no plans to make a paid version for this software.

2. Installing and Using Ururau

2.1How to get Ururau

The researchers of the software Ururau use the controller of versions for FOSS named Bitbucket. Bitbucket is a site for collaborative development, i.e., allows multiple developers to work on the same source code. Thus, the developer interested in cooperating registers himself, and the project manager allows the developer to submit the code changes to the site. However, anyone has access to the code without having to register. The process consists of installing the Mercurial (https://mercurial.selenic.com) and making the command "hg clone

https://bitbucket.org/ururau/ururau". Then, for the development and use of software Ururau, the user goes freely to http://ururau.ucam-campos.br. After downloading the zip file, he/she unzips it into a directory and runs ururau.jar. It is important to highlight that the Environment Java Runtime (JRE) version 6.0 or higher, which can be obtained in http://java.com/getjava, must be installed on the computer.

To build the model, using the source code, it is required a development environment - Integrated Development Environment (IDE), preferably NetBeans 7.0, which can be obtained in http://www.netbeans.org.

The environment of Ururau is internally composed by one or more process commands, which extends functionalities of the JSL. The file README.TXT, which presents an explanation on how to operate each command, can also be downloaded at the same address in the software.

2.2 Overview of the Ururau

Figure 1 presents an overview of the GUI of Ururau. As it can be seen in the illustration of an instant of programming, the model is constructed through the graphical interconnection of the software modules and of the further editing of the data relating to the logic and dynamics of the system.

Notice, in Figure 1, all parts of the GUI. The main menu is located at the top left of the screen. Just below, as found in most of today's software with graphical visualization, there is the toolbar, and also the modeling elements (modules). These elements have a set of functions for using in the development of the simulation model. A compact overview of the model is also presented at the bottom left, in case this does not fit the screen of the computer so that the modeler is able to view it as a whole. It can be seen, on the desktop, an example of a small model in preparation with the edit window open.

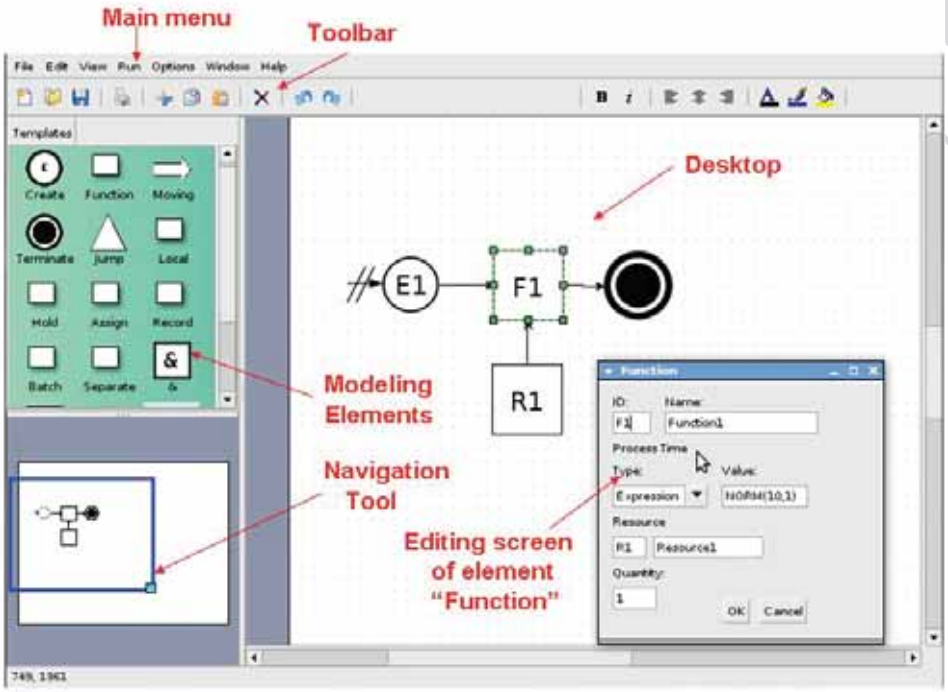


Figure 1: GUI of the Ururau.

For the construction of the simulation model, the modeler only has to drag and drop with the mouse the modeling elements for the desktop, according to the logic of the system under

development. After that, the editing module is made by clicking with the right mouse button on it to open the respective edit box of each element. The data system, as the times of events, logical and operational rules of the system and so on, can be inserted in the edit box.

The Ururau, still under development, shows, in Table 1, the following list of modules to develop models through the GUI.

Table 1: Module list to construct DES models.

Modules	Description
Entities Generator	Responsible for generating entities.
Batch	Makes a specified number of entities enter into a queue, being, then, transformed into one entity.
Hold	Holds the entities in a queue until a condition is met.
Separate	Undoes a temporary batch done by the Batch command.
Seize	Holds one or more entities for processing.
Release	Releases the requested resource for the entity.
Delay	Time that an entity takes to process.
Process	Aggregates, in a single command, commands Seize, Delay, Release, in that order.
Terminate	Ends the list of commands.
JumpTo	Diverts the processing of commands to another command.
Decide	Along with JumpTo, diverts the execution of commands depending on a condition. There are two types of decision: by chance and condition.
AddAtribute	Adds an attribute within the current entity.
AddVariable	Adds a variable in the model.
Record	Serves as a counter or measures the time interval, depending on an attribute (previously defined), with the value of the current simulation time.
Write	Writes results from variable, attribute or expression in a file while a simulation runs.

Table 2 shows the list of functions that can be used in the models developed with the Ururau with application examples.

Table 2: Examples of the use of probability distributions in Ururau

Distribution Functions	Example in Ururau
Exponential (mean)	EXPO(3)
Normal (mean; standard deviation)	NORM(5.1,1)
Lognormal (LN mean, LN standard deviation)	LOGN(3,0.2)
Triangular (minimum; most likely; maximum)	TRIA(1,5,10)
Weibull (form; scale)	WEIB(0.9, 0.1)
Poisson (mean)	POIS(2)
Beta (form; scale)	BETA(0.8, 0.2)
Uniform (minimum; maximum)	UNIF(3,8)

2.3 Applications with Ururau

The researchers who work on the project of Ururau software is currently involved with issues related to the following points:

(a) Comparative tests of DES models built in Ururau and other commercial simulators such as Arena and ProModel. In this topic, the work of Peixoto *et al*. (2013) presents some results achieved so far and shows that the Ururau allows execution of simulation models up to five times faster than commercial software with equivalent results.

(b) Research on new methods to be employed in the modeling and simulation of dynamic and stochastic systems with aspects, for example, of incorporation of Artificial Neural Networks to represent intelligent decisions in DES models. In this issue, the work of Silva *et al*. (2014) presents the results achieved so far.

(c) Use of the JSL library and Ururau software to be employed in the execution of simulation models of Monte Carlo. In this subject, the work of Peixoto, Rangel and Matias (2012) also presents the results obtained.

(d) Development of mechanisms and palettes in the GUI of Ururau software to enable integration of DES models with Programmable Logic Controllers. In this issue, the work of Cardoso *et al*. (2014) also presents the results obtained.

(e) Construction of DES models in Ururau considering discrete aspects associated with transport systems integrated with the continuous component of the emission of pollutant gases from a fleet of trucks. In this topic, Rangel and Cordeiro (2015) report the results achieved so far and show there is no direct relationship of proportionality between, for example, the delivery time of products and the total of the carbon monoxide emissions produced by vehicles of the logistics system.

An important characteristic that makes a FOSS different from a software of commercial simulation is that it allows users to create their own palettes of high-level objects to new applications. Objects of "high-level" are those that simulate the significant objects for a given class of models. For a traffic simulator, for example, the high-level objects could be different types of vehicles, a specific palette for a particular calculation that is repeated in many models of the same nature, or even another palette that can associate, for example, an Artificial Neural Network which represents a decision with a high degree of logical complexity of an operator of a system, among others.

3. Architecture and Resources of the Ururau

Figure 2 shows the architecture of the software Ururau. In all layers, we used the Java language. The uppermost layer is the GUI. This software layer deals with the conversion of the graph model, which comprises a directed graph to a sequence of commands of the Ururau core. The middle layer is the software core. This layer is composed of process commands as creation of entities, retention of resource (Seize), delay of time (Delay), resource release (Release), among other typical commands in DES software. The JSL library is in the lower part of the software. Then, the lower layer converts the model, which consists of a series of process commands to a sequence of discrete events to be performed during simulation. The simulation results of the lowermost layer and errors are passed again to the intermediate layer, which, in turn, passes again to GUI.

Even the Ururau software layers are interconnected; modeling can be performed on any of them. For example, in the first option, the simulation analyst can model using graphic language in the uppermost layer of Ururau. In the second choice, he can directly use the core of the Ururau and develop the model with the aid of a compiler, such as the integrated development environment (IDE) NetBeans 7.0.1. This core has a list of commands based on the own JSL. In the third and final option, the analyst still can program directly in JSL using an approach different from the Process View as object-oriented, explained in detail in Rossetti (2008).

The JSL is the engine of the software, and the integration of its components is illustrated in Figure 3. The reader can understand the software design in terms of diagrams of class at the link: https://bitbucket.org/ururau/ururau/downloads.

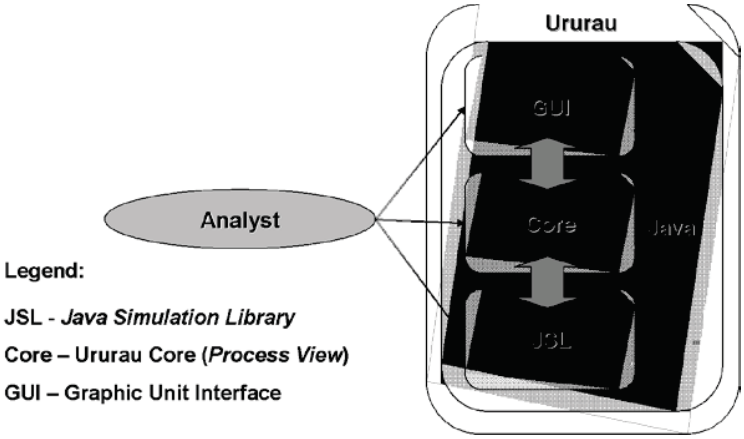


Figure 2: Architecture of the Ururau.

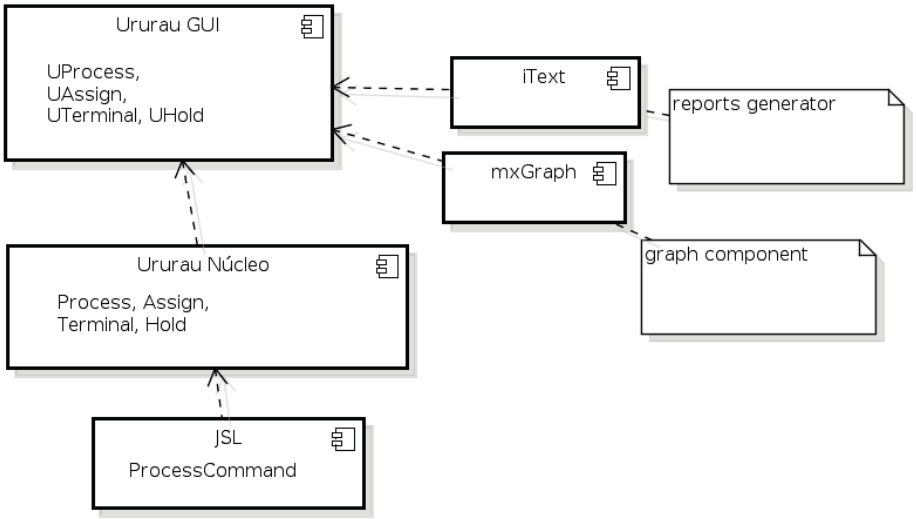


Figure 3: Diagram of the Ururau Components

3.1 Data Output and Reserved Words of the Ururau Software

Currently, there are two ways of data output with Ururau software. One is through reports and other through text file.

The generation of the reports is done with the aid of the iText library (http://itextpdf.com). This generation happens at the end of the simulation execution. It is necessary, a at a minimum, a PDF viewer in Windows, or the Evince, if the operating system is GNU / Linux. Appendix A presents a report generated after the execution of a simulation model in Ururau.

Now, it is necessary that the modeler uses the Write module and integrates it to the logic of the model through text file. In this case, that module must be in the right position of functioning in the model to, thus, be able to export to a file the right data to be analyzed. After running the simulation model, its data may be interpreted in a spreadsheet from the text file generated.

A reserved word of the Ururau is the variable TNOW, which returns the current time of the simulation in units of time of the model. To find out which is the number in the queue of some

function or location via graphical interface, it is necessary to follow its ID. For example, to evaluate the number in the queue in a XOR-type or Decisor of conditional type, simply place the expression "F1 <3" in the desired arc.

3.2 Communication of the Ururau with Programmable Logic Controllers (PLC)

The Ururau also allows communication with PLC. The simulation with control system is, usually, used for offline commissioning of control systems of industrial plants. This enables testing on control systems without stopping the plant.

The integration of Ururau with the OPC server is performed with the OPC client Utgard (http://openscada.org/projects/Utgard). This OPC client is part of the openSCADA software, which is a supervisory system, and of data acquisition open source, and is independent of platform. The Utgard was chosen because it is an OPC client open source, and 100% of the code is in Java, although interfacing with DCOM (Distributed Component Object Model), which uses native code of the operating system MS Windows© to access the OPC server. The OPC tags are like variables within the Ururau. They are set in an OPC server or OPC client, which makes a connection with the PLC. For the user, the access via GUI can be accomplished through the Tools menu and option OPC Communication. It is necessary to create a Windows user account with access permissions used for DCOM components of the OPC server chosen. The communication will be successful when, clicking 'Connect', the text box lists the predefined tags in the OPC server selected as shown in Figure 4.

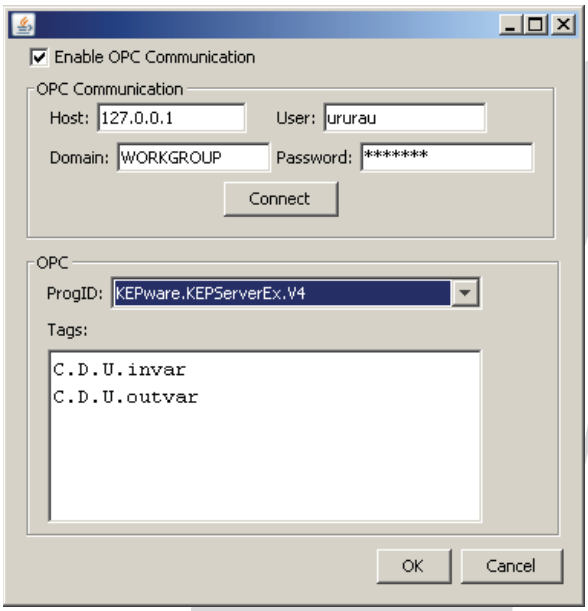


Figure 4: OPC Communication of the Ururau.

4. Example OF Model

The system used for the example model is hypothetical. It refers to a typical environment of an industrial production line and was used only as a way to illustrate the application of the software. Both the system and data related to processing times are based on information from training courses on a basic level in DES offered by Paragon. This company commercializes the Arena software in Latin America. The model addresses then a system of production line of a supposed auto parts factory.

4.1System description

The hypothetical system, illustrated in Figure 5, consists of a production cell with four workstations.

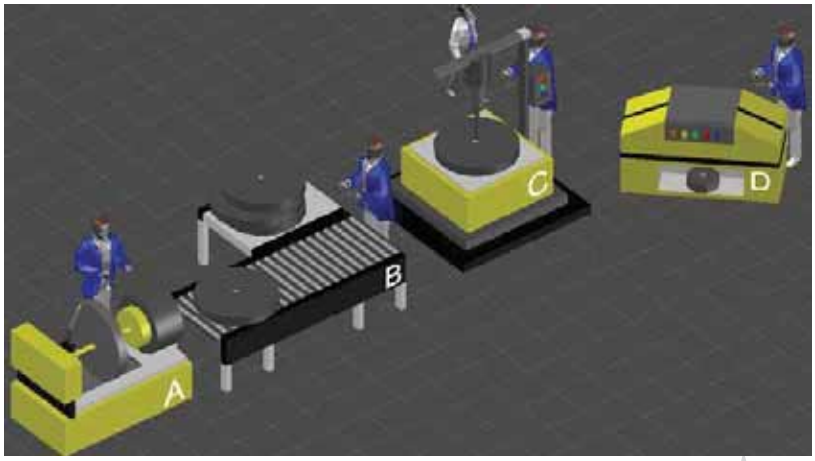


Figure 5: Illustration of an industrial production system, where (a) Lathe, (b) Inspection Station, (c) Drilling Machine and (d) Grinding Machine.

The first workstation is related to a Lathe, which process time follows a normal function with an average of 3 minutes and standard deviation of 1 minute. The second refers to an Inspection Station with a process of duration equivalent to a normal of mean of 2 minutes and standard deviation of 1 minute, being the this one performed by an operator. 10% of the parts are discarded because of quality problems. After the Inspection Station, there is a Drilling Machine that requires an operator to work with process time that follows a triangular of (2, 3 and 4.5) minutes. The parts that leave the Drilling Machine should only be taken for the Grinding Machine if its queue of parts in standby is less than 6. If the queue is greater than or equal to 6, the parts must be diverted to another line so as not to interrupt production. The operating time of the Drilling Machine follows a normal with a mean of 3 minutes and standard deviation of 1 minute. The arrival of the parts in the Cell of Production occurs according to an exponential function of mean of 3.5 minutes in the Lathe workstation.

4.2 Simulation Model

For the development of the simulation model, the methodology proposed by Banks (2010), with the following steps for this work, was used: problem formulation, construction of the conceptual model, translation of the conceptual model to Ururau software, verification and validation, and experimentation.

For the implementation of the simulation model, the parameters shown in Table 3 were adopted.

Table 3: General parameter of the model simulation round.

System Parameters	
Replication time	3000 minutes
Number of replications	150 times
Warm up	0 minutes

All data were arbitrary and serve only as background information for registering the parameters of the model round.

Figure 6 shows the simulation model of this system and Table 4 presents information related to system data.

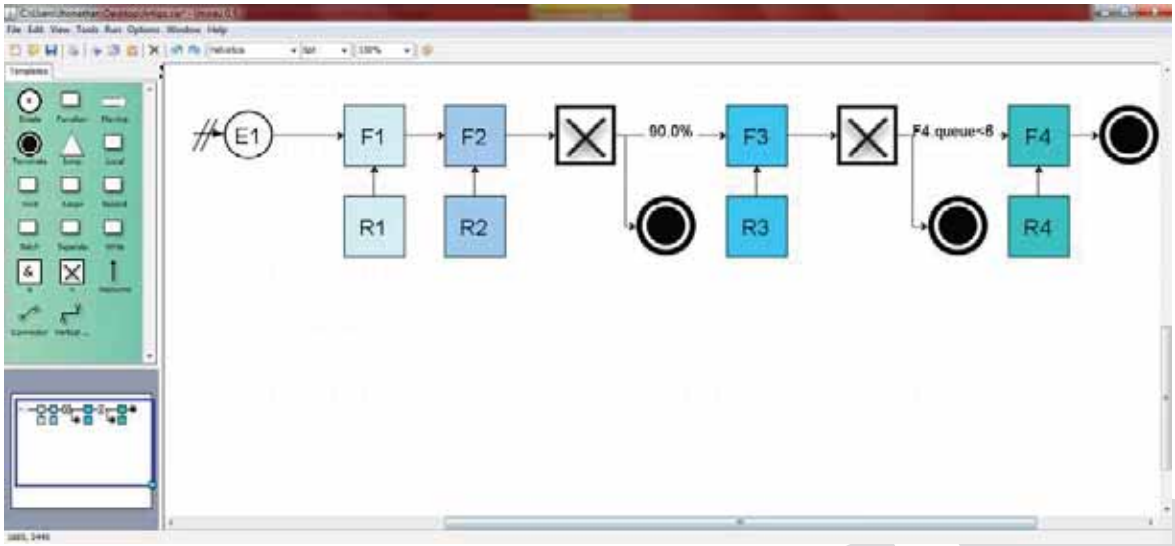


Figure 6: Simulation model of the productive cell.

The Ururau uses the same logic elements of the conceptual language IDEF-SIM proposed by Montevechi *et al* (2010) for DES projects. That is, once the conceptual language IDEF-SIM is figured out, the logic and operational rules of the simulation model built with Ururau software can be easily understood.

Table 4: Nomenclature of the functions presented in the modeled system.

	Description	Parameter
E1	Entity: Arrival of parts in process	Function: EXPO (3.5) per minute.
F1	Process: Process carried out in the industrial machine Mechanical Lathe	Function: NORM (3,1) per minute.
F2	Process: Process of manual inspection of parts	Function: NORM (2,1) per minute.
F3	Process: Process held in industrial machine drilling with the aid of an operator	Function: TRIA (2,3,4.5) per minute.
F4	Process: Process carried out in industrial machine grinding	Function: NORM (3,1) per minute.
R 1	Resource: Lathe	Mechanical resource
R 2	Resource: Inspection Station	Manual resource
R 3	Resource: Drilling Machine	Mechanical resource
R 4	Resource: Grinding Machine	Mechanical resource
C 1	Decisor: Decides the path of the part in the process regarding the size of the waiting queue for the industrial machine grinding	Analysis of the size of the queue: If (grinding machine. queue) < 6 the part goes to the grinding machine, if

		not, the part goes to other queue
G 1	Counter: Records the number of parts that have gone through all the workstations present in the manufacturing process.	Computational resource

The model was executed in 12h, characterizing it as a terminal system. The simulations were initiated only after the model has been full verified and validated by using the methodological guidelines proposed by Sargent (2013).

Table 5 shows the results of the main variables of interest of the simulation model. These data were extracted from the report that is generated when the execution of the simulation model is ended. In Appendix A, the report is seen with all the information that can be generated. Note that the size and, therefore, the amount of report information are directly proportional to the size and complexity of the simulation model.

Table 5: Simulation model data executed in Ururau.

Time in queue (min)					
	AV	SD	SI	Minimum	Maximum
Lathe	9,425	3,114	0,500	4,568	18,999
Inspection	0,311	0,052	0,008	0,193	0,492
Drilling Machine	2,592	0,655	0,105	1,439	5,066
Grinding Machine	1,011	0,270	0,043	0,528	2,080
Number in queue (un)					
	AV	SD	SI	Minimum	Maximum
Lathe	2,719	0,972	0,156	1,210	5,745
Inspection	0,089	0,016	0,003	0,055	0,155
Drilling Machine	0,668	0,186	0,030	0,336	1,439
Grinding Machine	0,260	0,075	0,012	0,130	0,554
Resource busy (%)					
	AV	SD	SI	Minimum	Maximum
Lathe	85,4	3,1	0,5	78,1	94,4
Inspection	57,2	2,2	0,4	51,6	63,9
Drilling Machine	81,1	3,0	0,5	73,2	90,7
Grinding Machine	76,7	3,0	0,5	67,1	85,2

5. Final Remarks

This paper presented the Ururau software and provided an overview of its features and how to use. We encourage readers interested in using it to visit the site in https://bitbucket.org/ururau.

The use of the Ururau can have the following advantages: firstly, the possibility to understand the source code and internal structure of an environment of DES. Secondly, the possibility of developing simulation models for research purposes, where specific algorithms may be added to the source code of the model, as an optimization algorithm, for example. Thirdly new features to the Ururau simulator can be added, in order to make it more compatible for some sort of specific approach and, thus, facilitate modeling the desired system.

Another issue, concerning the use of Ururau, regards the flexibility in developing models of simulation at any of the component layers of the structure of the Ururau software. This means that

models on the layer of the JSL (library of free simulation), in the core of the Ururau or top layer of the Ururau GUI, can be produced.

In the lower layer, models can be constructed by linking libraries in JSL to each other and, thereafter, be obtained a model of a system in Java code. The main advantage of this approach is the flexibility in constructing the simulation model. The disadvantage is the longer period for model development and the need to have an experienced programmer in Java in the construction of the simulation model. In the intermediate layer, Ururau Core, one can manipulate JSL libraries and, also, take advantage of the models developed for Ururau. At this point, the main profit is the possibility to develop other simulators for specific purposes. That is, if there is the necessity to create a simulator to analyze different models within a particular purpose, these models can be constructed quickly and accurately, without the modeler being a specialist in discrete simulation. Finally, models can be constructed graphically in the layer of the GUI of the Ururau.

As described in item 2.3, we have carried out some applications with this tool in undergraduate and graduate levels in modeling and simulation courses. The software has not been employed in high school classes yet. In the case of undergraduate level applications, the Ururau software proved to be a good opportunity for students of Computer Science to investigate the structure and source code from an environment to develop DES models. In relation to the graduate level, the software has been shown to be promising with students of Industrial Engineering. In this case, the research has been conducted in order to investigate new features and algorithms, such as the use of artificial neural networks to represent smart decisions in DES models We cannot say that the tool is more or less valuable than other similar ones. Our final aim with Ururau software is to help expand the simulation area beyond the restrictions of commercial software and proprietary code. Namely, enabling others to better understand the practical application and internal conception of the computational structure of the source code of software for the development of DES models.

ACKNOWLEDGMENTS

The authors thank the National Council for Scientific and Technological Development (CNPq) and the Research Foundation of the State of Rio de Janeiro (FAPERJ) for financial support for this research. They also thank Maria Marta Garcia for her assistance in translating the text into English.

REFERENCES

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation*. 5th ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc, 2010.

Cardoso, LD; Rangel, JJA; Nascimento, AC; Laurindo, QMG; Camacho, JC. (2014). Discrete event simulation for teaching in control systems. In: *Winter Simulation Conference*, Savanah. p. 3608-3617.

King DH and Harrison HS (2013). Open-Source Simulation Software "JAAMSIM". In: *Proceedings of the Winter Simulation Conference.*

Lawson, B. and Leemis, L. (2009). Simulation Fundamentals. *Proceedings of the 2009 Winter Simulation Conference.*

Montevechi JAB, Leal A, Pinho A, Costa RF, Oliveira ML and Silva AL (2010). Conceptual Modeling in Simulation Projects by mean adapted IDEF: an Application in a Brazilian company. In: *Proceedings of the Winter Simulation Conference.*

Rangel, JJA, and Cordeiro,ACA. (2015). Free and Open-Source Software for Sustainable Analysis in Logistics Systems Design. *Journal of Simulation (Print)*, v. 9, p. 27-42.

Peixoto, T. A.; Rangel, J. J. A.; Matias, I. O. (2012). Usando o JSL para Simulação de Monte Carlo. *GEPROS. Gestão da Produção, Operações e Sistemas (Online*), v. 4, p. 135-152.

Peixoto, T. A.; Rangel, J. J. A.; Matias. (2014). Free and Open-Source Simulation Software "Ururau". In: *Winter Simulation Conference*, Savannah, GA - USA. p: 1-2.

Peixoto, T. A.; Rangel, J. J. A.; Matias, I. O.; Montevechi, J. A. B.; Miranda, R. C. (2013). Ururau - Um Ambiente para Desenvolvimento de Modelos de Simulação a Eventos Discretos. *Pesquisa Operacional para o Desenvolvimento*, v. 5, p. 373-405.

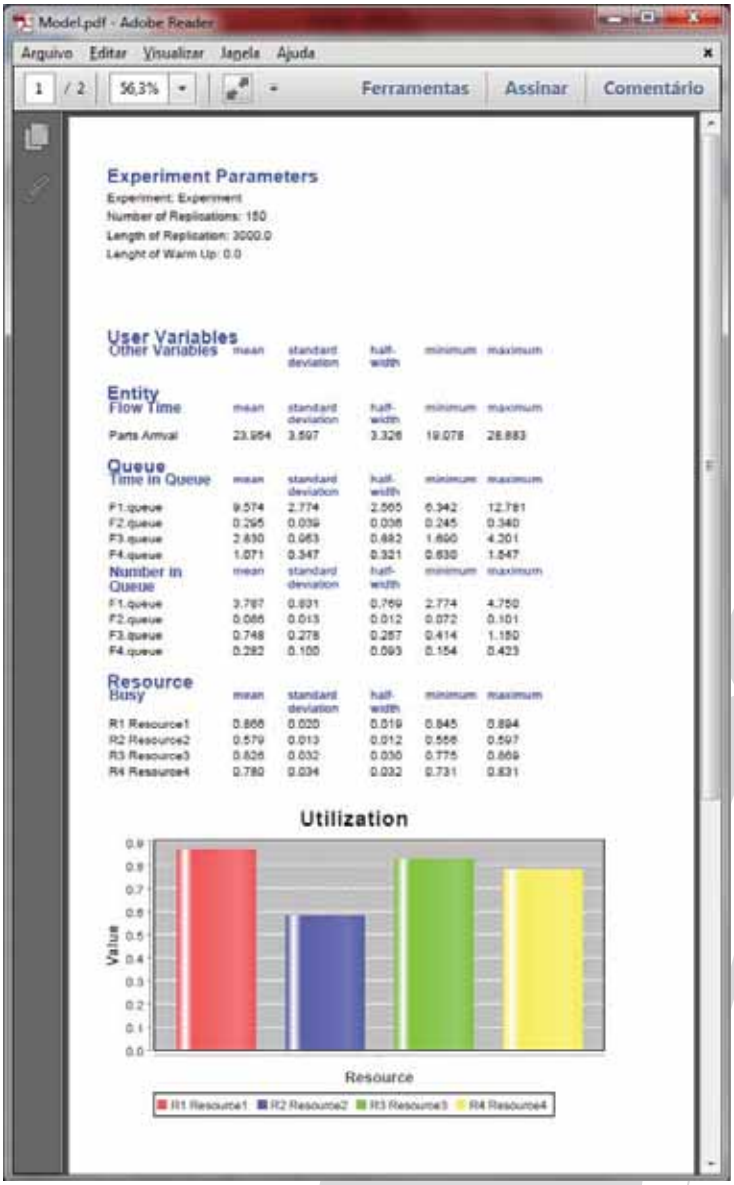
Rosseti, M.D. (2008) Java Simulation Library (JSL): an open-source object-oriented library for discrete-event simulation in Java. *Int. J. Simulation and Process Modeling*, Vol. 4, No. 1, p. 69-87.

Silva, M. G. D. ; Rangel, J. J. A. ; Silva, D. V. C. ; Peixoto, T. A. ; Matias, I. O. (2014). Decisão com Redes Neurais Artificiais em Modelos de Simulação a Eventos Discretos. *Pesquisa Operacional para o Desenvolvimento*, v. 6, p. 299-317.

Sargent, R. G. (2013) Verification and Validation of Simulation Models. *Journal of Simulation*, v. 7, p. 12-24.

Swain, J. J. (2007) Discrete Event Simulation Software: New Frontiers in Simulation, *OR/MS Today - INFORMS*, v. 34, n. 5, p.32-43.

Appendix A: Report of a simulation model in Ururau.


Model.pdf - Adobe Reader
Arquivo Editar Visualizar Janela Ajuda
1 / 2
56,3%
Ferramentas
Assinar
Comentário
Experiment Parameters
Experiment: Experiment
Number of Replications: 150
Length of Replication: 3000.0
Lenght of Warm Up: 0.0
User Variables
Other Variables mean standard deviation half-width minimum maximum
Entity
Flow Time mean standard deviation half-width minimum maximum
Parts Arrival 23.954 3.597 3.326 19.078 28.883
Queue
Time in Queue mean standard deviation half-width minimum maximum
F1.queue 9.574 2.774 2.565 6.342 12.781
F2.queue 0.295 0.039 0.036 0.245 0.340
F3.queue 2.830 0.963 0.882 1.600 4.301
F4.queue 1.071 0.347 0.321 0.630 1.647
Number in Queue mean standard deviation half-width minimum maximum
F1.queue 3.797 0.931 0.769 2.774 4.750
F2.queue 0.066 0.013 0.012 0.072 0.101
F3.queue 0.748 0.278 0.257 0.414 1.150
F4.queue 0.282 0.100 0.093 0.154 0.423
Resource
Busy mean standard deviation half-width minimum maximum
R1 Resource1 0.866 0.020 0.019 0.845 0.894
R2 Resource2 0.579 0.013 0.012 0.556 0.597
R3 Resource3 0.826 0.032 0.030 0.775 0.869
R4 Resource4 0.780 0.034 0.032 0.731 0.831
Utilization
Value
Resource
R1 Resource1 R2 Resource2 R3 Resource3 R4 Resource4