International Journal of Advanced Science and Technology
Vol.64 (2014), pp.11-20
http://dx.doi.org/10.14257/ijast.2014.64.02

Integration of Dynamic Resource Management in Discrete Event Models Using Arena Software

Nethal K. Jajo[1] and Kenan M. Matawie[2]

[1]*Research Portfolio, University of Sydney, Sydney, Australia*
[2]*School of Computing, Engineering and Mathematics, University of Western Sydney, Sydney, Australia*

[1]*Nethal.jajo@sydney.edu.au*, [2]*K.Matawie@uws.edu.au*

Abstract

Computer modelling of many real life systems is always challenged with how the dynamic changes of resources and queues are handled and integrated. In this paper, we integrate the dynamic changes of resources and queues in a discrete event model using Rockwell Software Arena, one of most used software tools in modelling discrete events, as it includes and considers the resources by queuing components both at fixed and predetermined level. The model proposed by the authors will extend this to the dynamic level by including the multiple queues that may require regrouping, verification and reallocation. The structure of this new model simply replaces all the external approaches to the management and synchronization of the resources by an internal/within the software integration structure. This model may also be considered and added to the Rockwell Arena Smart file library as a solution to dynamic resource management.

Keywords: *Arena, discrete dynamic simulation, nested search, queues, resources, smart model*

1. Introduction

In research, training pipeline, business, transport, educational course development and environmental sustainability, computer simulation models are ideally developed to represent the essential features of a real system. The more advanced and realistic the features of the model, the larger the contribution to determine the important factors affecting it, its sustainability and future continuity ([1] and [2]). For example, the goal of an analysis may be to reduce the average waiting time experienced by Marines while waiting for a particular formal training to commence [3]. In this case, and also in any education and training situation, one of the main and essential resources is the instructors. Arena software (Rockwell) provides an excellent tool in order to use the resources via the Resource, Queue, Set and Schedule modules in the Basic Process panel and the Advanced Set module in the Advanced Process panel. Recently, Taktak, Hachicha and Masmoudi [4] also showed how useful and essential Arena is for simulation optimisation techniques. However, these Arena features require the resources to be defined and determined in advance. The Spreadsheet and Resource module allows determining the characteristics of each Resource [5].

In various situations, particularly in the training environment, we are dealing with, a nested search (a multiple search for both queues and variable arrays) is needed along with a dynamical queuing system where the components (such as students and instructors) are interacting and their learning status is changing. Such an interaction may include the change

ISSN: 2005-4238 IJAST

from a trainee to a trainer, and possibly the vice versa. This type of interaction and update is known as a dynamic topology [6]. The Arena's features mentioned above are not suitable and adjusted to be used in such cases. Many authors, especially recently, such as Sacone and Siri [7], find it difficult to handle the dynamic resources and queues within Arena. [7] Introduced an external interface with Arena to handle this problem, however, their proposed solution was rather difficult and required optimization (maximizing the use of resources to minimize containers queue occupancy in the terminal) and it also required a decision on which application, software or tool can serve the interface better.

In this paper, we introduce an internal solution to this common problem, within Arena and without any need for any external interface, pausing the simulation and optimization. Thus, the proposed model dynamically handles all the resources and queues inside Arena. In learning environment, the proposed model is used as an example to help Arena models' users and developers to capture the dynamic change in the resource, queue and dynamic updates while the model is in progress and/or execution. No doubt this model can be part of the Arena Smart File Library, which provides a collection of models that demonstrate a variety of modelling techniques needed for today's applications and researches. Changes, new ideas, accuracy and amendments to models can be further captured when validating and verifying such models, since this paper's aim is not to discuss this, however, validation and verification were part of ongoing procedures associated with the development and integration of our proposed model. As mentioned in [6], it is essential to establish the desired level of accuracy between the model and the real system needs. Other important issues related to understanding, presenting, integrating and validating the model can be explored in Bertolino, De Angelis, Di Sandro and Sabetta [8].

2. The Problem

To illustrate the problem, let us consider the following typical example. We have to model a training pipeline with, say, three courses that are related to the students' level: course 1 for level 1 (assuming a class size of 40 students), course 2 for level 2 (with a class size of 30 students) and course 3 for level 3 (with a class size of 20 students). The enrolment proportions are 60%, 20% and 20% for course 1, 2 and 3, respectively. We also need to assume the yearly intake which is (say) 500 students distributed discretely as 40% level 1, 30% level 2 and 30% level 3. After the completion of these three courses some students can become instructors, and let us assume this is 5% of each of the three types of instructors. The rest of the graduated students will depart the model. There are three types of instructors associated with the courses with 5, 7 and 9 Years Of Service (YOS), respectively. Each instructor, in a specified type, is allowed only to instruct the course associated with that type. The initial number of all instructors is assumed to be, say, 50 distributed discretely as 33% type 1, 33% type 2 and 34% type 3. The ratio of instructors to students is assumed to be 10% for all levels. The instructor's attrition rate is another important input for such models; we assume it to follow a triangle distribution with parameter values as listed in Table 1.

A particular case for the above typical example is the training pipeline of pilots in both military and commercial airlines where a training pilot, after having completed certain courses and having gained the required experience, can be promoted to or certified as a qualified flying instructor. The reverse situation is when a qualified flying instructor for a certain level is attending, or required to attend, some higher or additional training course(s) and/or to gain experience.

The developed model, as described in the next section, will consider the dynamic changes in the learning status, the queue of the instructors, and the resource updates as

we are progressing with the training program. It is clear that these dynamic changes and updates are part of all the higher education institutes experience, plans and goals.

Table 1. The triangle distribution parameter values for the attrition rates of the instructors

Instructor Type	Minimum Value	Most Likely Value	Maximum Value
1	80	85	90
2	50	55	60
3	50	55	65

3. Model Development

As mentioned earlier, this model will be developed through the Arena software tool, and illustrated the example in Section 2. The formulation of the conceptual model begins with the determination of the model boundaries. The model boundaries for this example are given in Figure 1. The following notations will be used throughout the paper. $\{CT_q\}$ represents the set of taken courses. If a course i is taken then $ct_i = 1$, and $ct_i = 0$ if this course is not taken. $\{AC_q\}$ represents the set of assigned courses. $\{CE_q\}$ represents the set of course enrolments. $\{CC_q\}$ represents the set of course capacities. I_q represents the set of instructor types (each corresponding to a distinct queue) and $\{I_{qr}\}$ represents the set of instructor types and ranks in a queue. The first index q represents the instructor queue while the second index r represents the instructor rank within that queue.

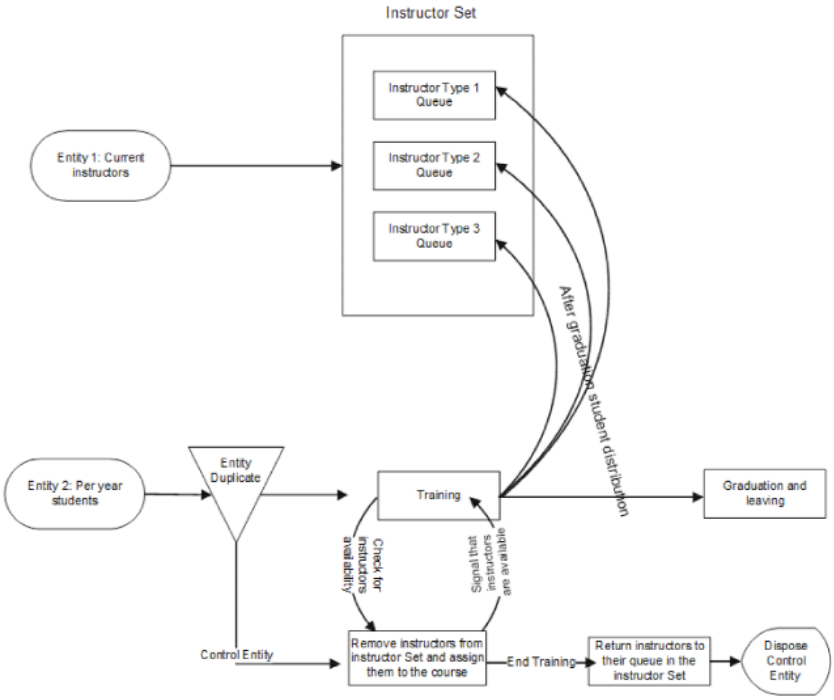

Figure 1. Dynamic Changes of Resources and Queues Model Boundaries.

3.1. Instructor distribution and queuing

Current instructors will be distributed based on their type. Using the Assign module from the Block panel, each instructor type will be assigned a referencing label. Instructors will be held in three different queues using the Advanced Set and the Hold Modules in the Advanced Process panel. These modules can be viewed in Figure 2.

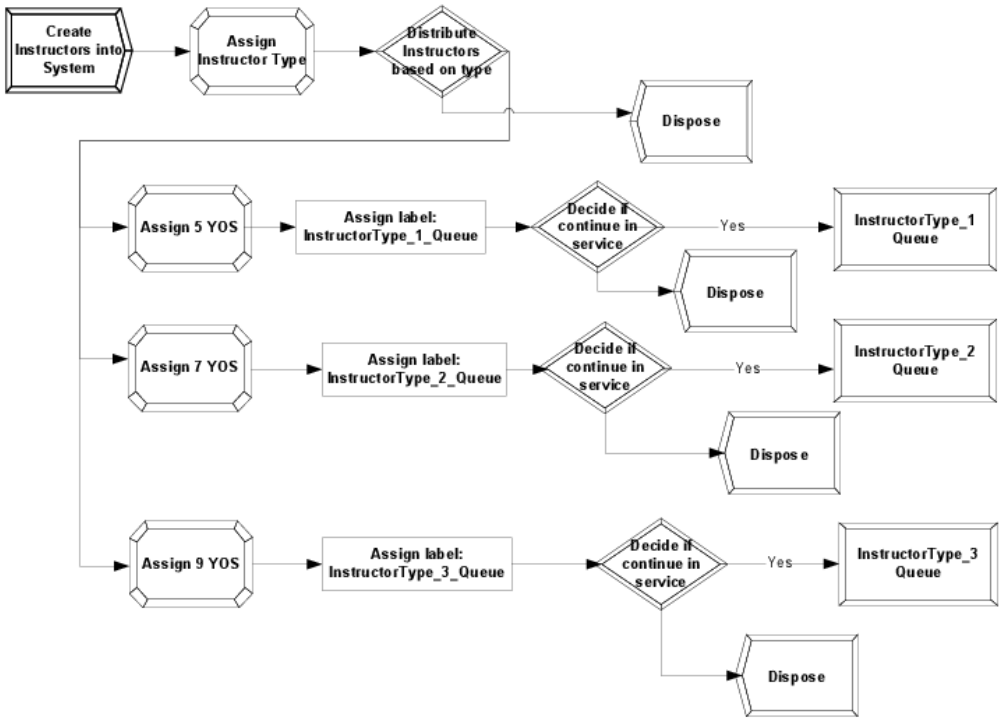

Figure 2. Instructor Distribution and Queuing model.

The set will be called Instructor Set $\{I_{qr}\}$ where q=1, 2, …, Q represents both the queue and the instructor type. Q is the maximum number of instructor queues or types. In our case study, Q=3. r=1, 2, …, R represents the rank of an instructor within a queue. R is the total number of instructors in a specified queue. In our case study, we used the Symbol Number of Queue To Search variable to represent the value R.

3.2. Student dynamic flow

For each student, the model will search for the required courses to complete. If the model finds a course that has not been completed by a student and the course capacity has not been reached, then the student is assigned to that course and will wait until the instructors are available. A control-entity will be created by duplicating the student entity. This control-entity will check the availability of instructors in numbers and type of that course. When the sufficient numbers of the specified instructors are available, then the student will be enrolled in the course for a period of time determined by the length of the course. In other words, each student will be checked according to the mathematical iteration process given in Equation (1).

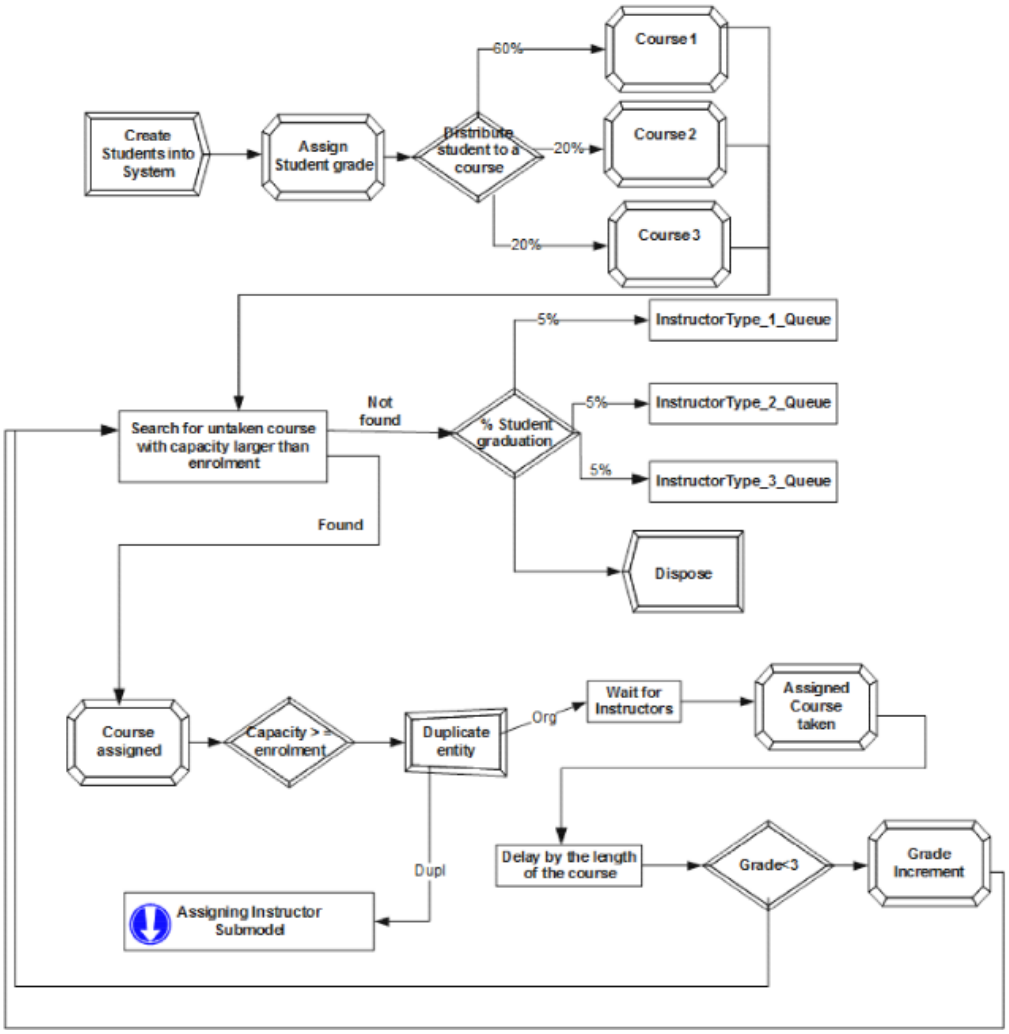

Figure 3. Student dynamic flow

For $i = 1, 2, \ldots, Q$
For $ct_i \neq 1$ and $cc_i > ce_i$ → student is assigned to ac_i
If number of $I_i \geq cc_i \times 10\%$
$ce_i = ce_i + 1$
$ct_i = 1$ (1)
Next i

At the end of each course, instructors will return to their queues while the students will be checked for their progress and further courses required for graduation. When graduated, 15% of the student will be appointed as instructors; that is the 5% for each instructor type will be moved to the specified instructor queues and will be assigned 0 YOS by using the Assign Module from the Block panel. Figure 3 illustrates this in details.

3.3. Instructor dynamic change, assignment and flow

The control-entity will search, assign and remove the required instructor from its queue. After the completion of the course the instructor entity will be returned to its queue and the control-entity will be disposed. As detailed in Figure 4, we start with assigning value zero to variable "QueueToSearch", and then it will be incremented by one by the Assign Module. This variable will be used to insure that all queues have been searched prior to finding another instructor. Another indicator variable will be assigned to the queue identity of the selected (matched) instructor. This variable is denoted by "SymbolNumberOfQueueToSearch" (introduced at the end of the Section 3.1) and its value will be given by the MEMBER(InstructorSet, QueueToSearch) function. This MEMBER function is an Arena Set variable function, which returns the construct number of a particular set member. In other words, this function will return the number of instructors queuing in each queue within { I_{qr} }.

When "SymbolNumberOfQueueToSearch" is registering over zero, which means there are instructors queuing, the model then will start searching to occupy and match instructors to relevant courses.

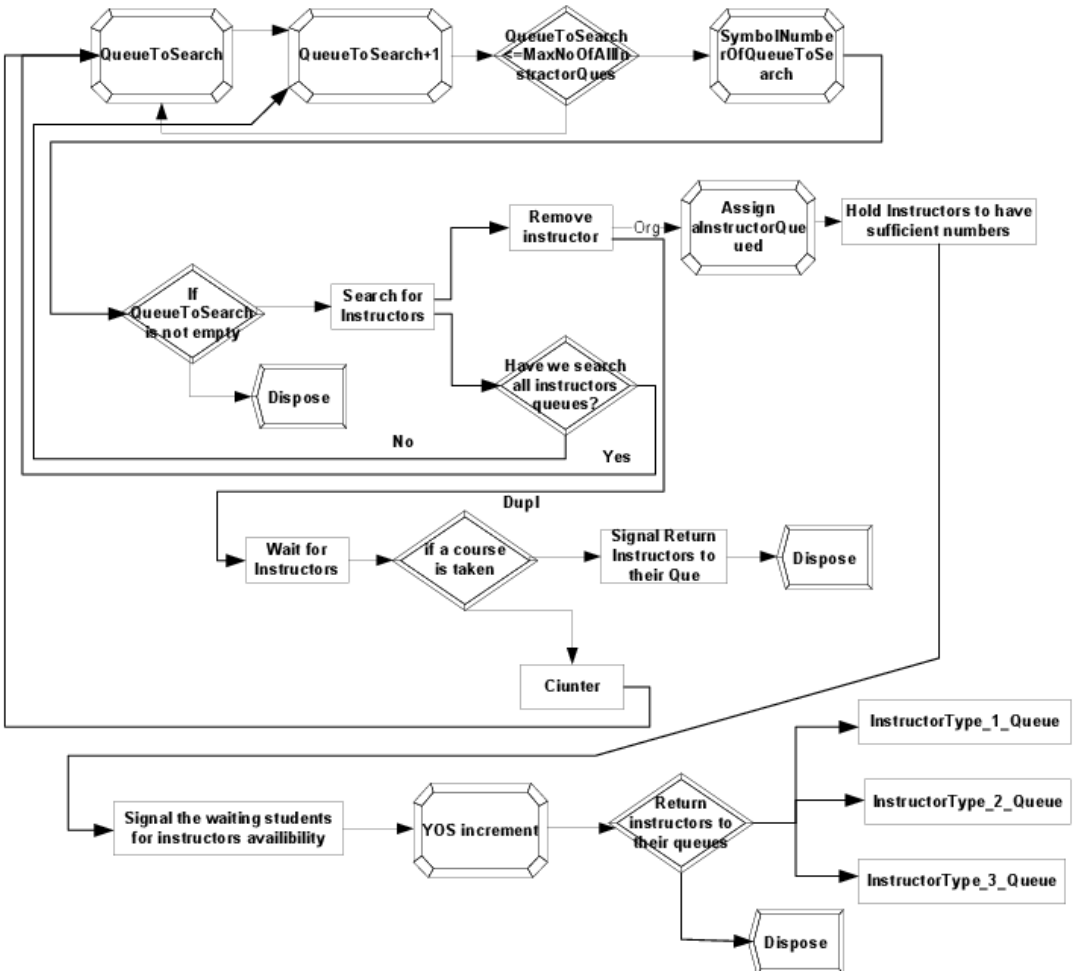

Figure 4. Assigning instructor submodel

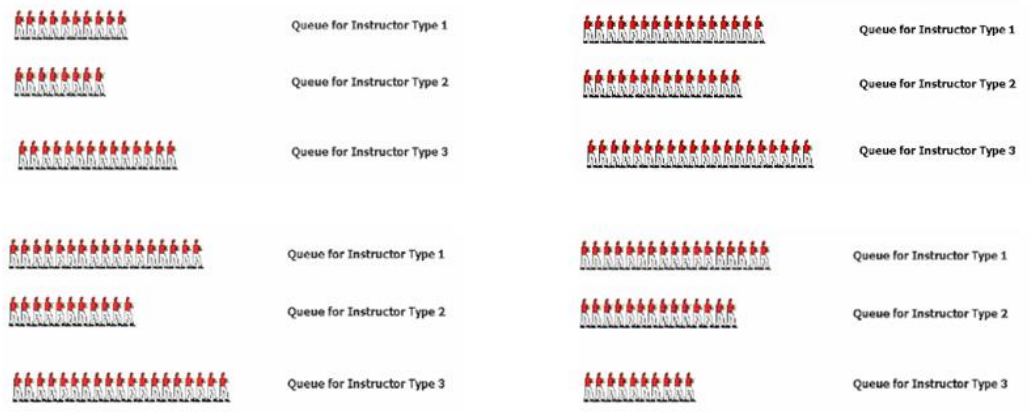

Figure 5. Four sets of instructors in four different model running times

```
For QueueToSearch = 1 to Q
    R = MEMBER({I_qr}, QueueToSearch)
      For J = 1 to R
        If AQUE(SymbolNumberOfQueueToSearch, J, NSYM(I_q)) = NSYM(ac_q);
        remove the instructor with rank J from SymbolNumberOfQueueToSearch queue;
        InstructorQueued = QueueToSearch;
        NoOfMatchedInstructors = NoOfMatchedInstructors + 1;
        If NoOfMatchedInstructors ≥ InstructorsRatio × ce_q;
        run the course and return the instructors to their queue;
    Next J                                                          (2)
Next QueueToSearch
```

The condition for this nested search, Equation 2, will be based on using the Arena Set variable function named NSYM and the Arena queue variable function named AQUE. The NSYM function was used to translate the attribute name "InstructorType" into its numerical value. Then the AQUE function was used to check-and-match both the course assigned and the instructor type. This function will return the numerical value of the required instructor type attribute of the entity at the j^{th} rank in "SymbolNumberOfQueueToSearch" queue. If the numerical value of the AQUE function matches the numerical value of the assigned course, then the Remove module from the Blocks panel will remove the matched instructor of rank *J* from queue "SymbolNumberOfQueueToSearch". The removed instructor will be assigned an attribute named "InstructorQueued" with a value equal to that of variable "QueueToSearch". This attribute value will be used to return the instructors to their original queue in the "InstructorSet" at the completion of the course. This search and remove of instructors from their queues will continue till the specified required number of instructors is met. Then, a signal will be sent to start the course, and instructors will be returned to their original queue in the "InstructorSet" at the end of the course. We used the "Next Label" feature in the Assign module from Blocks panel to send back the removed instructors to their original labelled queue. At this stage, the control-entity will be disposed. Figure 4 illustrates the assigning of instructors to the courses. The model was animated using different pictures and colours, for the student entity an uncoloured human picture was chosen, a red man picture was chosen for the instructor entity, and a document picture for the control-entity. Three animated queues for the instructors were placed on the model window to show the dynamic change in their counts. This animation can be viewed while the model is running and when the animation view mode

is switched on. Figure 5 represents four sets of instructor dynamic changes in four different times arbitrarily selected to show the effectiveness of this model in capturing the dynamic change in the number of instructors during the model execution.

4. Conclusion

Computer modelling is an effective way to simulate the important factors affecting the real life environment. As we mentioned earlier, this is true in every domain, particularly in a training pipeline, when designing an education course, and analysing its sustainability and future continuity, an example which was used here to demonstrate the proposed model.

In this paper the dynamic changes in the queuing resources were modelled using Arena 13.0. This work showed the model development and the Arena model improvement to handle important issues of dynamic management and changes to the resources inside the software, a problem that challenged most of the researchers and modellers and was bypassed through external means. Nested search and dynamic queue change techniques were illustrated using a typical training/education example. This situation is commonly encountered when using Arena to model training pipelines or educational training courses.

The details of the proposed model can be used for both training and as a reference tool to assist in model-building efforts. It is also important to mention that this paper is better understood when reading the model details, variables and structure provided in the graphs and even more clear when running the whole model. Incorporating a dynamic environment into a software tool, particularly into Arena, is a new approach, and an extremely useful one for both researchers and software users. As this model development is a new idea, not many references are available in this area yet.

References

[1] M. Jahangirian, T. Eldabi, A. Naseer, L. K. Stergioulas and T. Young, "Simulation in manufacturing and business: A review", European Journal of Operational Research, **(2010)**, pp. 203**.**
[2] F. T. S. Chan and H. K. Chan, "A comprehensive survey and future trend of simulation study on FMS scheduling", Journal of Intelligent Manufacturing, vol. 15, no. 1, **(2004).**
[3] J. Davenport, C. Neu, W. Smith and S. Heath, "Using discrete event simulation to examine Marine training at the Marine corps communication-electronics School", Proceedings of the 2007 Winter Simulation Conference, Washington DC, USA, **(2007)**.
[4] S. Taktak, W. Hachicha and F. Masmoudi, "A Computer-assisted Performance Analysis and Optimization (CPAO) of Manufacturing Systems based on ARENA® Software", International Journal of Advanced Science and Technology, vol. 39, **(2012).**
[5] W. D. Kelton, R.P. Sadowski and D.T. Sturrock, "Simulation with Arena", 4th edition, McGraw-Hill, New York, (**2007**).
[6] U. W. Pooch and J. A. Wall, "Discrete Event Simulation: A practical Approach", 4th edition, CRC press, Florida, (**2000**).
[7] S. Sacone and S. Siri, "An integrated simulation-optimization framework for the operational planning of seaport container terminals", Mathematical and Computer Modelling of Dynamical Systems, vol. 15, no. 3, (**2009**).
[8] A. Bertolino, G. De Angelis, A. Di Sandro and A. Sabetta, "Is my model right? Let me ask the expert", The Journal of Systems and Software, doi:10.1016/j.jss.2011.01.054, (**2011**).

Authors

Dr. Nethal K. Jajo

Dr. Nethal K. Jajo is Modelling and Projection Analyst at the University of Sydney, Australia. He has a PhD degree in Mathematical Statistics and Probability Theory with professional training in Discrete Event Simulation and System Dynamics Modelling. His industrial and academic research activities include: Data analysis, data mining, partial least squares path modelling, regression analysis and dynamic simulation.

Dr. Kenan M. Matawie

Dr. Kenan M. Matawie is a Senior Lecturer of Statistics in the School of Computing, Engineering and Mathematics at University of Western Sydney. His research interest is in application of statistics, particularly in statistical modelling, advance data analysis and developing statistical methods motivated by real-world problems. Kenan has a PhD from University of New South Wales, Australia.

